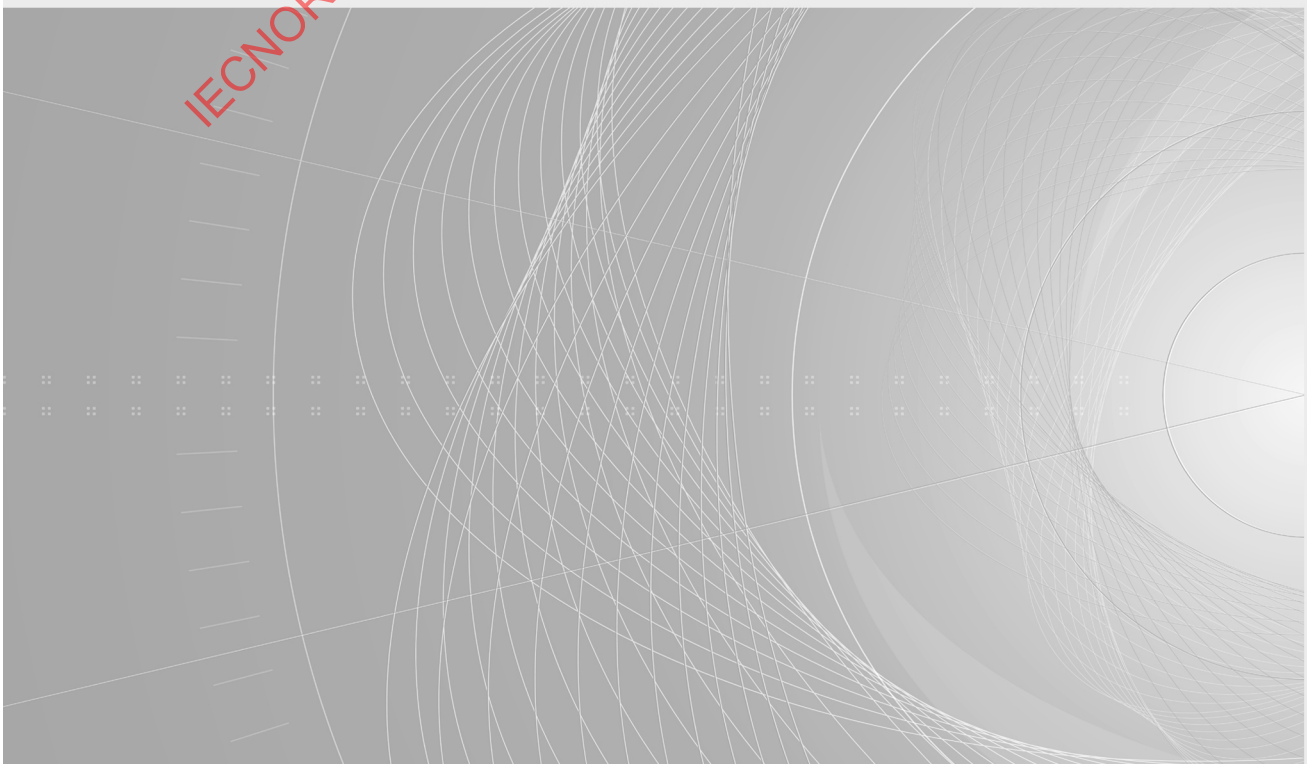
IEC 62605

Edition 3.0 2021-08

INTERNATIONAL STANDARD

Multimedia systems and equipment – Multimedia e-publishing and e-books – Interchange format for e-dictionaries



IEC 62605:2021-08(en)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, …). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC online collection - oc.iec.ch
Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 18 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC 62605

Edition 3.0 2021-08

INTERNATIONAL STANDARD

Multimedia systems and equipment – Multimedia e-publishing and e-books – Interchange format for e-dictionaries

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 33.160.60; 35.240.20; 35.240.30

ISBN 978-2-8322-9917-3

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

MULTIMEDIA SYSTEMS AND EQUIPMENT – MULTIMEDIA E-PUBLISHING AND E-BOOKS – INTERCHANGE FORMAT FOR E-DICTIONARIES

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

International Standard IEC 62605 has been prepared by technical area 10: Multimedia e-publishing and e-book technologies, of IEC technical committee 100: Audio, video and multimedia systems and equipment.

This third edition cancels and replaces the second edition published in 2016. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

The format described in Annex B was significantly enhanced in the following ways:

a) elements to enhance uses of data (e.g. web-related usage);
b) attribute for accessibility-related functions;
c) attributes for more detailed markup for data reusability.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
100/3589/FDIS	100/3602/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/standardsdev/publications.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

INTRODUCTION

Markets for multimedia e-books and e-publishing require standardization of formats for e-book data interchange among associated people, authors, data preparers, publishers and readers. The formats are classified into submission format, interchange format and reader's format. The submission format supports an interaction between authors and data preparers. The reader's format depends on e-publishing equipment. The interchange format provides an interchange format for data preparers and publishers and, therefore, should be e-publishing equipment independent.

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of a patent. IEC takes no position concerning the evidence, validity, and scope of this patent right.

The holder of this patent right has assured IEC that s/he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with IEC. Information may be obtained from the patent database available at http://patents.iec.ch.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those in the patent database. IEC shall not be held responsible for identifying any or all such patent rights.

MULTIMEDIA SYSTEMS AND EQUIPMENT – MULTIMEDIA E-PUBLISHING AND E-BOOKS – INTERCHANGE FORMAT FOR E-DICTIONARIES

1 Scope

This document specifies the interchange format for e-dictionaries among publishers, content creators and manufacturers.

This document does not address the following aspects:

- data formats for reading devices;
- elements necessary for final print reproduction only;
- rendering issues related to physical devices;
- security issues such as DRM for documents.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC TS 62229, *Multimedia systems and equipment – Multimedia e-publishing and e-book – Conceptual model for multimedia e-publishing*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
manufacturer
organization or person that manufactures hardware and/or software of the e-book

4 Position and requirements for interchange format for e-dictionaries

4.1 Interchange format for e-dictionaries in contents creation/distribution model

The conceptual model for multimedia e-publishing (IEC TS 62229) defines a contents creation/distribution model shown in Figure 1.

Author <--(1)--> **Data preparer** <--(2)--> **Publisher** --(3)--> **Reader**

IEC

Key

(1) content data in submission format

(2) content data in interchange format

(3) content data in reader's format

Figure 1 – Contents creation/distribution model

It should be noted that the role of manufacturers of e-dictionary hardware and software overlaps that of the publisher in Figure 1. Therefore, a slightly modified model will be assumed for this International Standard, as shown in Figure 2.

Author <--(1)--> **Data preparer** <--(2)--> **Publisher (manufacturer)** --(3)--> **Reader**

IEC

Figure 2 – Contents creation/distribution model (modified)

This document specifies the interchange format between data preparers and publishers, i.e. a format for (2) in Figure 2, though it may be used as a reader's format.

4.2 Requirements for interchange format for e-dictionaries

An interchange format for e-dictionaries needs to address the following.

- Description of keywords, links from the keywords to entries (link data) and the order of the entries.
- Description of articles for each entry (entry data). This includes text, image, and other multimedia functionalities generally required for e-books.
- Description of bibliographical data and other data. This should include the name of the author and the publisher, the title of the content and the explanatory note. The relationship between these concepts is visually represented in Figure 3.
- Description of contents written in various languages.

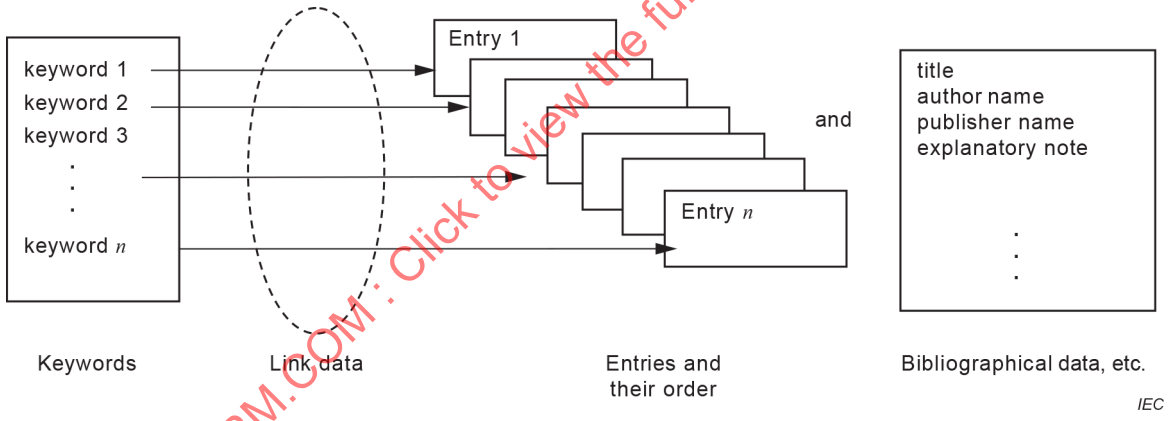


Figure 3 – Relationship between concepts

5 File formats

This document defines two XML-based formats. One is based on XMDF (as described in IEC 62448:2017, Annex B) and LeXML[1]. The format is hereafter called XMDF-LeXML format. The other is based solely on LeXML 3.0. They are presented in Annex A and Annex B, respectively.

[1] LeXML is the trade name of a product supplied by Digital ASSIST Ltd. Its original specifications are found at http://www.d-assist.com/index.html (in Japanese). This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

6 Semantics

Elements of the XMDF-LeXML format can be rendered in accordance with appropriate style specifications, which are outside the scope of this document.

Annex A
(normative)

XMDF-LeXML format

A.1 General

The XMDF-LeXML format is an interchange format for e-dictionaries multimedia e-book data interchange, targeted at data preparers and publishers rather than the reader, with an emphasis on mobile devices as a target platform. Much like HTML, this format does not split the document in fixed pages, but determines the layout according to the viewer device's display size, the font in use, and so on. In this annex, such contents will be referred to as "flowing content", as opposed to "paged content".

A.2 Overview of the format's structure

Flowing contents are usually composed of several concatenated flows. This annex makes no particular requirement concerning the way the flowing content should be split into individual flows. This decision is left to the data preparer, to accommodate the various types of content. For instance, a newspaper can have one flow per article, a novel one per chapter, and so on. It is also possible not to split the content, and to have only one flow. However, it should be noted that particularly large flows, or an extremely large number of flows, can impact on runtime performance, depending on the specific version of the viewer in use, the available memory, and so on.

The XML tree structure of the format is shown in Figure A.1.

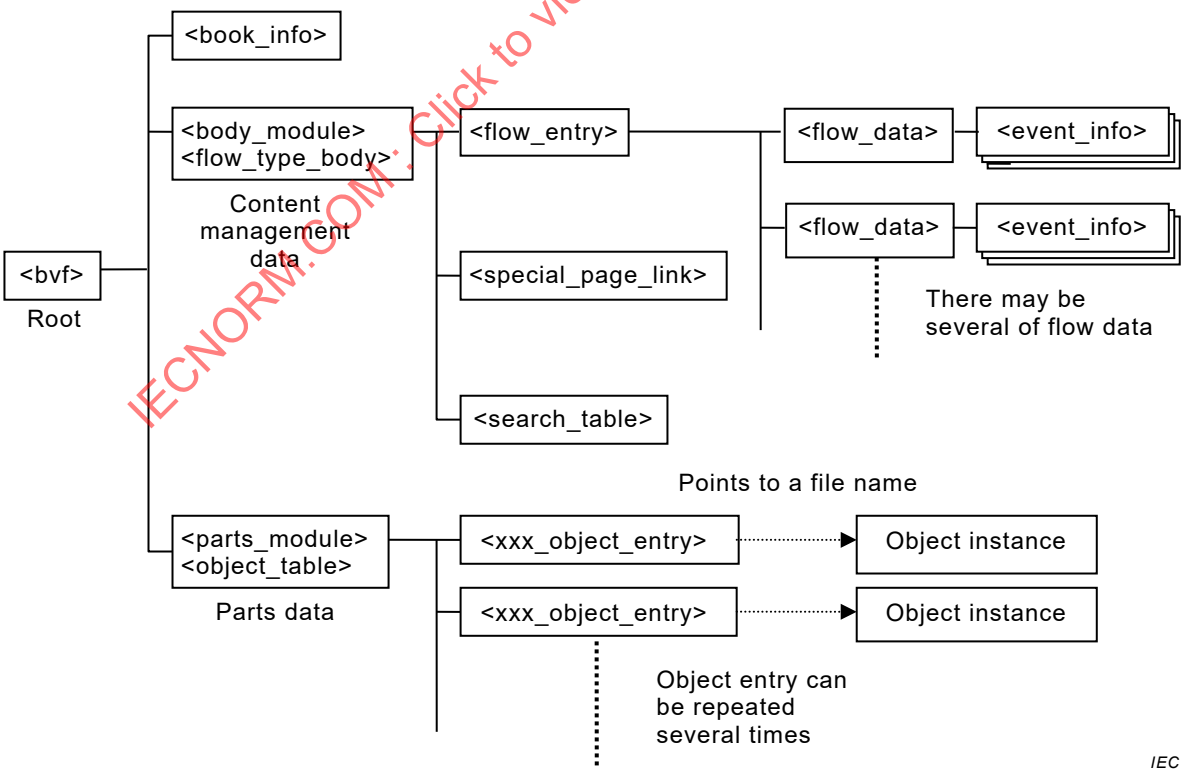


Figure A.1 – XML tree structure

The actual contents of each flow, in other words, what will be displayed by the viewer, is recorded in the object instance. The object instance is registered in object_entry, and associated with an ID number and other auxiliary data, turning it into playable/displayable data. Flow_data determines its content by pointing at such registered objects. In addition, information on functionalities such as page link is recorded in event_info.

The main part of Annex A is generic, and may be used for any country and language. However, some parts may have language-specific behaviour. Localization-related issues are detailed in Clause A.6.

A.3 Elements and attributes

A.3.1 General

The different types of values that may be used in the various elements or attributes are explained below. The elements and attributes detailed below will be valid throughout this annex, and will be referenced by other constructs. In the following explanations, alphanumeric characters refer to numerals from 0 to 9 and alphabetic letters from a to z and A to Z.

A.3.2 Page_ID

Page_ID specifies a unique identification number for the flow data of the flowing contents. It is a string starting with the characters "PG", followed by alphanumeric characters.

An example is shown in Figure A.2.

```
<flow_data flow_id="PG0002" ... />
```

Figure A.2 – Example of Page_ID

A.3.3 Object_ID

Object_ID specifies a unique identification number for objects used in the flowing contents. It is a string starting with the characters "OB", followed by alphanumeric characters.

An example is shown in Figure A.3.

```
<dynamic_text_object_entry id="OB0ue4".../>
```

Figure A.3 – Example of Object ID

A.3.4 Char_ID

Char_ID specifies an identification number for positions (character strings, etc.) within text and dictionary data objects. It is an alphanumeric string which is to be given uniquely in the text (see A.4.6.2) and dictionary data object instance (see A.4.6.3). Char IDs with the same value in different object instances are regarded as separate and don't affect each other.

An example is shown in Figure A.4.

```
<trigger_pointer id="OB29s0/CR0de4"/>
Click<char_id char_id="CR0001">here</char_id>for details.
```

Figure A.4 – Examples of Char_ID

A.3.5 Reading

For sorting purposes, reading can be useful to specify the reading of each word. Restricting the characters allowed for this purpose to a limited set makes it easier to define the sorting method. Such characters should be determined on a per language basis. All languages can use the characters listed in Table A.1 as a common base, while the localization (see Clause A.6) will describe the language specific extensions to it.

Table A.1 – Base characters for reading

Name	Corresponding characters [a]
Basic alphabet	A to Z (U+0041 to U+005A) a to z (U+0061 to U+007A)
Numerals	0 to 9 (U+0030 to U+0039)
Others	space (U+0020), ((U+0028),) (U+0029)
[a] [Attributes]ll values are in Unicode.	

An example is shown in Figure A.5.

```
<title reading="PI">π</title>
```

Figure A.5 – Example of reading

A.3.6 Filename

File names should be written using the following convention. The path is relative to the file in which this reference is made. Network paths should not be used. For portability concerns, it is recommended that only ACSII characters be used. Both the slash and backslash characters are acceptable as directory separators. It is also recommended not to use excessively long file names, as those might not be supported by the host operating system. Such notation for a file name is hereafter referred to as "Filename" in Annex A.

An example is shown in Figure A.6.

```
<dynamic_text_object_entry src="sect1.xml" type="text/x-bvf-text" id="OB0ue4"/>
```

Figure A.6 – Example of filename

A.3.7 Standard character

The standard character set of the document, as set by the default_ccs attribute of the <bvf> element (see A.4.2), is to be chosen from a well defined list, so as to ease the development of viewing software. However, this list may change for different localized versions of the XMDF-LeXML format. Any e-book data has to define its standard character set as one of or a combination of character set(s) listed in Table A.2 and those defined for a specific localization (see A.6.2).

Table A.2 – Standard character set

Character set name	Description
"ISO 646-IRV"	Characters in the range of US-ASCII

A.3.8 Standard character string

A string composed of standard characters is called a "standard character string". Unless specified otherwise, the spacing characters (space (U+0020), linefeed (U+000D, U+000A, U+000D+U+000A), tabulation (U+0009)) are to be handled as follows:

- space (U+0020) to be displayed as is;
- linefeed (U+000D, U+000A, U+000D+U+000A) not to be displayed, but simply ignored;
- tabulation (U+0009) to be displayed as if it were a single space.

Furthermore, because of restriction in the XML format, linefeeds (U+000D, U+000A, U+000D+U+000A) and tabulations (U+0009) in attribute values should be replaced by spaces when converting to the distribution format.

A.3.9 Extended character

Characters that have Unicode code points while not being among those listed below are referred to as "extended characters":

- standard characters;
- surrogate pair range (U+D800 to U+DFFF);
- BOM (byte order mark) (U+FFFE,U+FEFF);
- NON CHARACTER (U+FFFF);
- control characters (characters between U+0000 and U+001F except tabulation (U+0009) and linefeed (U+000A, U+000D), as well as DEL (U+007F)).

If an e-book indeed uses any extended character in its data, the name of a character set that covers those extended characters should be appended to the default_ccs attribute of the <bvf> element. Note that all extended characters used in the document do not need to be covered by the same character set, as it is possible to specify several character sets.

A.3.10 Extended character string

A string composed of standard characters and extended characters is called an "extended character string". Unless specified otherwise, the spacing characters (space (U+0020), linefeed (U+000D, U+000A, U+000D+U+000A), tabulation (U+0009)) are to be handled the same way as for standard character strings.

A.3.11 External character

To display a character that is neither a standard character nor an extended character, it is possible to use the <external_char> element described below.

<external_char> inserts an external character. The viewer may display it in accordance with the following methods:

a) display the character set by the alt_set and alt_code attributes;

b) display the image set by the alt_img or alt_vimg attributes;

c) display the alternative letter set by the alt attribute.

Its syntax is given in Relax NG compact format below and explained in the following text.

NOTE For definitions that appear in the Relax NG compact representation of each element in this annex, see Clause A.8.

```
external_char = element external_char { attlist_external_char, text }

attlist_external_char &=
```

attribute alt_set { text }?,

attribute alt_code { text }?,

attribute alt_img { text }?,

attribute alt_vimg { text }?,

attribute img_type { text }?,

attribute alt { text }?

[Attributes]

alt_set: Together with the alt_code attribute, it allows designating the External character to be used. This alt_set attribute indicates the font name, while the alt_code attribute indicates the character code point within the font. The alt_set attribute is written in the following way:

alt_set = "font1,font2, ..."

The alt_set attribute may hold several font names, separated by "," (U+002C). In that case, the viewer should use the first font of the list that is available (either from the platform, or included in the contents data itself) to display the character.

alt_code: Selects a character code point in the font specified by the alt_set attribute. It may be written either as a decimal number or a hexadecimal number, prefixed by "0x". In case several fonts have been defined in the alt_set attribute, the character code shall represent the same character in all of them. This attribute can be omitted.

alt_img: Defines an alternative character image. Written as a Filename. Before opening the file indicated by this attribute, the img_type attribute should be checked for authorized file types. Note that it may be used only when <external_char> is used in a text object instance. When both alt_img and alt_vimg are used, the file types shall match. This can be omitted. When this attribute is set, display should be done in accordance with the following methods:

a) Monochrome images

Black pixels represent the letter, and white pixels the background. The font colour and background colour are to be displayed according to the colour attribute of the <font> element.

b) Images with levels of grey

Black pixels represent the letter, and white pixels the background. The font colour and background colour are to be displayed according to the colour attribute of the <font> element. The colour of "grey" pixels shall be computed as an intermediate value between the font colour and the background colour.

c) Colour images

Displayed as is.

alt_vimg: Defines an alternative character image to be used when the text is displayed vertically (as can be the case in some languages, such as Japanese). Written as a Filename. When omitted, the image defined in the alt_img attribute should be used both for horizontal and vertical layout. Before opening the file indicated by this attribute, the img_type attribute should be checked for authorized file types. Note that it may be used only when <external_char> is used in a text object instance. When both alt_img and alt_vimg are used, the file types shall match. This can be omitted. When this attribute is set, the display should be done in accordance with the same methods as alt_img.

img_type: Defines the mime type of the images files set in the alt_img and alt_vimg attributes. Currently, only PNG and JPEG are supported, and should be written as:

"image/png"

"image/jpeg"

When either alt_img, alt_vimg or both are set, this attribute is required. As with these two attributes, it can only be used in an <external_char> element inside a text object instance.

alt: Alternative character string. Written as a standard character string. May be omitted.

Examples are shown in Figure A.7.

```
<external_char alt_set="oooextchars" alt_code="47268" alt_img="ou.img"
                    alt_vimg="ou_v.img" img_type="image/jpeg" alt="鴎"/>
<external_char alt="間"/>
<external_char alt_set="sharp_extchars" alt_code="0x2345" alt="間"/>
```

Figure A.7 – Examples of the external_char element

A.3.12 External character string

An external character string is a string composed of standard characters, external characters, or both. Unless specified otherwise, spacing characters (space (U+0020), linefeed (U+000D, U+000A), tabulation (U+0009)) should be handled the same way as they are handled in standard character strings.

Examples are shown in Figure A.8.

```
森<external_char alt_img="ou.png" alt_type="image/png" alt="鴎"/>外
内田百<external_char alt_set="sharp_extchars" alt_code="0x2345" alt="間"/>
```

Figure A.8 – Examples of external character string

A.3.13 External extended character string

An external extended character string is a string of standard characters, extended characters, external characters, or a combination of any of the above. Unless specified otherwise, spacing characters (space (U+0020), linefeed (U+000D, U+000A), tabulation (U+0009)) should be handled the same way as they are handled in standard character strings.

A.3.14 Coordinates

Data type to be used to store coordinates, dimension and other similar information composed of an x and a y value. It is written as "(x, y)". The name of the attribute that uses this type of data depends on the element.

The coordinate system explained below will be used in this annex. The origin is at the top left corner, the x-axis oriented rightwards, and the y-axis downwards. As the system of coordinates used by the viewer to map things on the screen is implementation-dependent, it will not be discussed here.

Local coordinate system:

The coordinate system local to an object takes its origin in the upper left corner of the circumscribed rectangle, and has the same orientation as the general coordinate system. Positions within an object should be expressed in the local coordinate system.

Examples are shown in Figure A.9.

```
<vertex position="(100,200)"/>
```

Figure A.9 – Examples of coordinates

A.3.15 Polygonal_region

Data format to store the apexes of a polygon, or any other ordered sequence of vertexes. Each vertex is stored in a <vertex> element. When defining the shape of a polygon, the edges shall not cross. If they do, the viewer's behaviour is unspecified. Its syntax is given in Relax NG compact format below and explained in the following text.

```
vertex = element vertex { attlist_vertex, empty }

attlist_vertex &= attribute position { text }
```

[Attribute]

position: the position of the apex, expressed as Coordinates. This attribute shall not be omitted.

Examples are shown in Figure A.10.

```
<vertex position="(100,0)"/>
<vertex position="(0,100)"/>
<vertex position="(200,100)"/>
```

Figure A.10 – Examples of the vertex element

A.3.16 Colour

Data type to define colors. The following attributes are defined.

[Attributes]

color_space: Specifies the colour space to be used. Currently, only RGB is accepted. If this attribute is omitted, the viewer should act as if RGB was set.

color: Specifies the colour name. Colour names or numerical values may be used. Acceptable colour names are listed in Table A.6. The default value depends on the actual element and context. Numerical values are to be written in the following syntax.

In RGB: written as #RRGGBB. With RR, GG, BB being hexadecimal numbers, ranging from 00 to FF. Grayscale pixel values are represented by setting RR, GG and BB to the same value.

opacity: level of opacity. Ranging from 0 (transparent) to 100 (opaque). Presently, the only admitted value is 100, and if the attribute is omitted, it defaults to 100.

Examples are shown in Figure A.11.

```
<font color="#FF0000"/>
                    <!-- the color_space is unspecified, and thus defaults to RBG-->
<font color="#FF0000" opacity="100"/>
<font color="black"/>
```

Figure A.11 – Examples of colour

A.3.17 Date

Data format to store dates. It uses the same representation as ISO 8601. For instance, 1994-11-05T08:15:30-05:00 corresponds to November 5, 1994, 8:15:30 am, US Eastern Standard Time. Abbreviated forms are also accepted. Please refer to http://www.w3.org/TR/NOTE-datetime for details.

An example is shown in Figure A.12.

```
<publication_date type="publish">1994</publication_date>
```

Figure A.12 – Example of date

A.3.18 Time

Data format to specify durations. Written as "XXdXXhXXmXXsXXXms", where X stands for a digit between 0 and 9. For instance, "10d5h30m10s015ms" would mean 10 days, 5 hours, 30 minutes, 10 seconds and 15 milliseconds. Abbreviated forms such as "5m30ms" or "1s" are possible. There is no upper bound to the number of days.

An example is shown in Figure A.13.

```
<flip_animation renewal_time="1s" >
```

Figure A.13 – Example of time

A.3.19 Country

Data format to specify a country name. Written according to the ISO 3166-1 alpha 3 standard, in lower case.

An example is shown in Figure A.14.

```
<publication_place>jpn</publication_place>
```

Figure A.14 – Example of Country

A.3.20 Personal_name

Data format to store people's names, such as the document author. It is stored under the <personal_name> element. Its syntax is given in Relax NG compact format below and explained in the following text. Several child elements are used to define the various parts of the name: first name, middle name and last name. This information shall be entered via the elements defined below. At least one of <first_name>, <middle_name>, and <last_name> shall be specified.

```
personal_name =
  element personal_name {
    attlist_personal_name,
    ((first_name?, middle_name?, last_name?)
     | (first_name?, last_name?, middle_name?)
     | (last_name?, first_name?, middle_name?)
     | (last_name?, middle_name?, first_name?))
  }
attlist_personal_name &= empty
```

[Child elements]

<first_name> Sets the first name. Written as an external character string. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
first_name = element first_name { attlist_first_name, TextWithGaiji }
attlist_first_name &= attribute reading { text }?
```

[Attribute]

reading: Pronunciation of the first name, written as a reading. May be omitted.

<middle_name> Sets the middle name. Written as an external character string. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
middle_name = element middle_name { attlist_middle_name, TextWithGaiji }
attlist_middle_name &= attribute reading { text }?
```

[Attribute]

reading: Pronunciation of the middle name, written as a reading. May be omitted.

<last_name> Sets the last name. Written as an external character string. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
last_name = element last_name { attlist_last_name, TextWithGaiji }
attlist_last_name &= attribute reading { text }?
```

[Attribute]

reading: Pronunciation of the last name, written as a reading. May be omitted.

An example is shown in Figure A.15, storing the name "John Smith."

```
<personal_name>
        <last_name>Smith</last_name>
        <first_name>John</first_name>
</personal_name>
```

Figure A.15 – Example of the personal_name element with its descendants

A.3.21 Organization_name

Data format to define company's name, such as publishers. It is stored in the <organization_name> element. Written as an external character string. Its syntax is given in Relax NG compact format below and explained in the following text.

```
organization_name =

  element organization_name { attlist_organization_name, TextWithGaiji }

attlist_organization_name &= attribute reading { text }?
```

[Attribute]

reading: Pronunciation of the company's name, written as a reading. May be omitted.

An example is shown in Figure A.16.

```
<organization_name >ABCD Corporation</organization_name>
```

Figure A.16 – Example of the organization_name element

A.3.22 Address

Data format used to define an address, telephone number, email, and other information. Address-related information is all stored in child elements of a main <address_info> element. Its syntax is given in Relax NG compact format below and explained in the following text.

```
address_info =
  element address_info {
    attlist_address_info,
    postal_code?,
    address?,
    telephone?,
    fax?,
```

```
    mail_address?,
    website?,
    address_other_info?
  }
attlist_address_info &= empty
```

[Child elements]

<postal_code> Stores the postal code (zip code) as a standard character string. May be omitted. Its syntax is given in Relax NG compact format below.

```
postal_code = element postal_code { attlist_postal_code, text }
attlist_postal_code &= empty
```

<address> Stores the address as an external character string. May be omitted. Its syntax is given in Relax NG compact format below.

```
address = element address { attlist_address, TextWithGaiji }
attlist_address &= empty
```

<telephone> Stores the phone number as a standard character string. May be omitted. Its syntax is given in Relax NG compact format below.

```
telephone = element telephone { attlist_telephone, text }
attlist_telephone &= empty
```

<fax> Stores the fax number as a standard character string. May be omitted. Its syntax is given in Relax NG compact format below.

```
fax = element fax { attlist_fax, text }
attlist_fax &= empty
```

<mail_address> Stores the mail address as a standard character string. May be omitted. Its syntax is given in Relax NG compact format below.

```
mail_address = element mail_address { attlist_mail_address, text }
attlist_mail_address &= empty
```

<website> Stores the home page's URI as a standard character string. May be omitted. Its syntax is given in Relax NG compact format below.

```
website = element website { attlist_website, text }
attlist_website &= empty
```

<address_other_info> Allows storing additional information not covered by the preceding elements as an external character string. May be omitted. If this information is to be displayed, spaces (U+0020) and linefeeds (U+000D, U+000A, U+000D+U+000A) should be displayed as is, while tabulations (U+0009) should be displayed as spaces. Its syntax is given in Relax NG compact format below.

```
address_other_info =
  element address_other_info {
    attlist_address_other_info, TextWithGaiji
  }
attlist_address_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
```

An example is shown in Figure A.17.

```
<address_info>
        <postal_code>75008</postal_code>
        <address>xxx avenue des Champs-Élysées, Paris, France </address>
        <telephone>01 53 xx xx xx</telephone>
        <mail_address>xxx@XXXXX.fr</mail_address>
        <website>http://www.XXXXX.fr</website>
</address_info>
```

Figure A.17 – Example of the address_info element with its descendants

A.3.23 Permission

Sets the permissions, such as the right to print, or copy. The various permissions are stored in child elements of the <permission_info> element. When the <permission_info> element is omitted, all permissions are set to the same value as when each permission element is omitted. The following explanations refer to an "authenticated user". Under usual circumstances, all viewers of the document are considered authenticated. However, the distribution format may include DRM technologies and authentication mechanisms. The syntax of the <permission_info> element is given in Relax NG compact format below and explained in the following text.

```
permission_info =
  element permission_info {
    attlist_permission_info, print_permission?, copy_permission?
  }
attlist_permission_info &= empty
```

[Child elements]

<print_permission> Defines whether printing is permitted or not. When omitted, it is to be considered as if the permission attribute was set to "no". Its syntax is given in Relax NG compact format below and explained in the following text.

```
print_permission =
  element print_permission { attlist_print_permission, empty }
attlist_print_permission &=
  [ a:defaultValue = "no" ]
  attribute permission { "authorized" | "no" }?
```

[Attribute]

permission: Defines whether printing is permitted or not. The following values are possible.

"authorized": authenticated users may print.

"no": no one can print (default value).

<copy_permission> Define whether copying is permitted or not. When omitted, it is to be considered as if the permission attribute was set to no. Its syntax is given in Relax NG compact format below and explained in the following text.

```
copy_permission =
  element copy_permission { attlist_copy_permission, empty }
attlist_copy_permission &=
  [ a:defaultValue = "no" ]
  attribute permission { "authorized" | "in_device_only" | "no" }?
```

[Attribute]

permission: Defines whether copying is permitted or not. The following values are possible.

"authorized": authenticated users may copy.

"in_device_only": authenticated users may copy, but only within the viewer device. If the device does not provide mechanisms to prevent external copy, then copy is forbidden.

"no": No one can copy (default value).

An example is shown in Figure A.18.

```
<permission_info>
        <print_permission permission="authorized"/>
</permission_info>
```

Figure A.18 – Example of the permission info element with its descendant

A.3.24 Keyword

With the <keyword_list> element, it is possible to attach a list of keywords to the bibliographical data, to a flow data, or to an object (in the present document, it is limited to bibliographical data). A <keyword_list> should contain one or more <keyword> elements as child element, each of these recording one keyword. The syntax of the <keyword> element is given in Relax NG compact format below and explained in the following text.

```
keyword = element keyword { attlist_keyword, TextWithGaiji }
attlist_keyword &=
  attribute category { text }?,
  attribute reading { text }?
```

[Attributes]

category: Defines the category the keyword belongs to, written as a standard character string. May be omitted.

reading: Records the pronunciation of the keyword, written as a reading. May be omitted.

[Child elements]

External character string: Records the actual keyword.

Examples are shown in Figure A.19.

```
<keyword_list>
        <keyword category="History" >Renaissance</keyword>
        <keyword>xml</keyword>
</keyword_list>
```

Figure A.19 – Examples of the keyword_list element with its descendants

A.3.25 Telephone_number

Data format to record telephone numbers. It allows dialing such a number when the viewer device is a telephone. It is written as a combination of the characters listed in Table A.3. The number of characters shall be between 1 and 64, inclusive.

If a function defined by the phone number cannot be executed, the viewer should not make the call. Namely,

- when the phone number string is too long for the device to handle,
- when the phone number string includes characters not listed in Table A.3,
- when the phone number string contains a function that cannot be executed.

Table A.3 – Usable characters for a telephone number

Character	ASCII code	Meaning
0 to 9	0x30 to 0x39	number
#	0x23	# button
*	0x2A	* button
-	0x2D	Ignored.
,	0x2C	Pause (one second). If a one second pause cannot be made, wait for key press instead.
/	0x2F	Pause (wait for key press).
P or p	0x50 or 0x70	Pause (wait for signal). If it is not possible to wait for the signal, wait for a key press instead.
+	0x2B	Sign to make an international call. Only at the beginning of a phone number. (If entered in another position, do not make the call.)

A.3.26 Mail_address

Data format to store an email address. Written as local-part@domain. local-part and domain may use any of the characters recorded in Table A.4. The maximum length for local-part@domain is 256 bytes. Within the following characters, "&" (0x26) shall be written as an xml entity: "&". Such xml entities are counted as one byte.

Table A.4 – Characters usable for email addresses

Category	Characters	ASCII code
numerals	0 to 9	0x30 to 0x39
alphabet	A to Z a to z	0x41 to 0x5A 0x61 to 0x7A
!	!	0x21
$	$	0x24
%	%	0x25
&	&	0x26
*	*	0x2A
+	+	0x2B
-	-	0x2D
.	.	0x2E
/	/	0x2F
=	=	0x3D
?	?	0x3F
^	^	0x5E
_	_	0x5F
~	~	0x7E

A.4 Description format details

A.4.1 General

Each book takes the form of an XML document as shown in Figure A.20.

```
<?xml version="1.0" encoding="UTF-8" ?>

<bvf id="1234" id-type="...">

        <book_info>
         ...              <!-- Bibliography comes here -->
        </book_info>

        <body_module>
        ...               <!--Flow data comes here -->
        </body_module>

        <parts_module>
          ...             <!-- Objects are registered here -->
        </parts_module>
</bvf>
```

Figure A.20 – A book XML document

The character encoding used in the document is specified in the usual XML way, with the encoding attribute of the <xml> element. This annex recommends using UTF-8 or UTF-16 to avoid conversion problems; see A.6.8 for further details.

A.4.2 Book information modules <bvf>

A.4.2.1 General

The book information module is recorded in the <bvf> element. It serves as a root element for all data in the book, all information related to the document are stored inside it (it may also happen that only the filename of external files is stored here). The syntax of the <bvf> element is given in Relax NG compact format below and explained in the following text.

```
bvf = element bvf { attlist_bvf, book_info, body_module, parts_module }
attlist_bvf &=
  attribute id_type { text }?,
  attribute id { text }?,
  attribute default_ccs { text },
  attribute display_size { text }?
```

[Attributes]

id_type:	Defines what type of number is stored in the id attribute. Written as a standard character string. May be omitted.
id:	Records the Identification number of this book, in the system specified by the id_type attribute. Written as a standard character string. May be omitted. Not to be confused with Reference ID (see A.4.6.3.4).
default_ccs:	Sets the name of the character set of the Standard characters (see A.3.7) and extended characters used in this annex (see A.3.9). When more than one character set is specified, they are separated by a "," (U+002C). Shall not be omitted.
display_size:	Specifies the display (screen) size that was assumed while creating the contents, written in the standard coordinates format. May be omitted.

[Child elements]

<book_info> Records bibliographical data. Shall not be omitted. See A.4.2.2 for details.

<body_module> Content management module. Shall not be omitted. See A.4.3 for details.

<parts_module> Parts data modules. Shall not be omitted. See A.4.5 for details.

An example is shown in Figure A.21.

```
<bvf id_type="ISBN" id ="xxx-x-xxxx-xxxx-x" default_ccs="ISO 646-IRV">
        <book_info> ... </book_info>           <!-- Bibliography -->
        <body_module> ... </body_module>  <!-- Content module -->
        <parts_module> ... </parts_module>  <!-- Parts data modules -->
</bvf>
```

Figure A.21 – Example of the bvf element with its descendants

A.4.2.2 Bibliographical data <book_info>

This is where bibliographical data, such as the author or the title, is stored. The syntax of the <book_info> element is given in Relax NG compact format below and explained in the following text.

```
book_info =
  element book_info {
    attlist_book_info,
    title_info,
    author_info?,
    publisher_info?,
    seller_info?,
    book_id_info?,
    classification_info?,
    rating?,
    publication_place?,
    publication_date_info?,
    net_price_info?,
    book_abstract?,
    front_cover_image?,
    spine_cover_image?,
    keyword_list?,
    other_book_info?
  }
attlist_book_info &= empty
title_info =
  element title_info {
    attlist_title_info,
    series_title?,
```

```
    title,
    subtitle?,
    edition_info?,
    title_other_info?
  }
```

[Child elements]

<title_info> Stores the information related to the title. Shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
title_info =
  element title_info {
    attlist_title_info,
    series_title?,
    title,
    subtitle?,
    edition_info?,
    title_other_info?
  }
attlist_title_info &= empty
```

[Child elements]

<series_title> The title of the series is recorded as an external character string in this element. If there is no series' title, it may be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
series_title =
  element series_title { attlist_series_title, TextWithGaiji }
attlist_series_title &= attribute reading { text }?
```

[Attribute]

reading: gives the pronunciation of the series' title as a reading. May be omitted.

<title> The title is recorded as an external character string in this element. It shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
title = element title { attlist_title, TextWithGaiji }
attlist_title &= attribute reading { text }?
```

[Attribute]

reading: gives the pronunciation of the title as a reading. May be omitted.

<subtitle> The subtitle is recorded as an external character string in this element. If there is no subtitle, it may be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
subtitle = element subtitle { attlist_subtitle, TextWithGaiji }
attlist_subtitle &= attribute reading { text }?
```

[Attribute]

reading: gives the pronunciation of the subtitle as a reading. May be omitted.

<edition_info> Information concerning the revision history of the book is recorded as an External character string in this element. May be omitted. If the information is to be displayed, the space character (U+0020), and the linefeed and carriage return characters (U+000D, U+000A, U+000D+U+000A) are to be displayed as is, but the tabulation character (U+0009) is to be displayed as if it were a space. Its syntax is given in Relax NG compact format below and explained in the following text.

```
edition_info =
  element edition_info { attlist_edition_info, TextWithGaiji }
attlist_edition_info &=
  attribute this_version { text }?,
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
```

[Attribute]

this_version: Specifies which of the cited versions is the present one. For instance, "Third revision". May be omitted.

<title_other_info> Other information related to the title may be stored as an External character string in this element. May be omitted. If the information is to be displayed, the space character (U+0020), and the linefeed and carriage return characters (U+000D, U+000A, U+000D+U+000A) are to be displayed as is, but the tabulation character (U+0009) is to be displayed as if it were a space. Its syntax is given in Relax NG compact format below.

```
attlist_title_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
```

<author_info> Stores the information related to the author. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
author_info = element author_info { attlist_author_info, author+ }
attlist_author_info &= empty
```

[Child elements]

<author> Registers each author. Its syntax is given in Relax NG compact format below and explained in the following text.

```
author =
  element author {
    attlist_author,
    (personal_name | organization_name),
    address_info?,
    author_other_info?
  }
attlist_author &=
  [ a:defaultValue = "author" ]
  attribute role {
    "author"
    | "editor"
    | "translator"
    | "supervisor"
    | "designer"
    | "photographer"
    | "illustrator"
    | "binder"
    | "planner"
    | "other"
  }?
```

[Attribute]

role: Defines the role of the person mentioned. The possible values are listed below. If omitted, it defaults to "author".

"author", "editor", "translator", "supervisor", "designer", "photographer", "illustrator", "binder", "planner", "other"

[Child elements]

<personal_name> / <organization_name>

Records the author's name in one of these two elements, according to whether the author is an individual or an organization, respectively written as a personal_name or organization_name.

The details of the <personal_name> and <organization_name> elements are given in A.3.20 and A.3.21, respectively.

<address_info> Address of the author, written in the Address data format. May be omitted. The details of the <address_info> element are given in A.3.22.

<author_other_info> Other information related to the author may be stored as an external character string in this element. May be omitted. If the information is to be displayed, the space character (U+0020), and the linefeed and carriage return characters (U+000D, U+000A, U+000D+U+000A) are to be displayed as is, but the tabulation character (U+0009) is to be displayed as if it were a space. Its syntax is given in Relax NG compact format below.

```
author_other_info =
  element author_other_info { attlist_author_other_info, TextWithGaiji }
attlist_author_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
```

<publisher_info> Stores the information related to the publisher. May be omitted. The <publisher_info> element has the following child elements. If <publisher_info> is specified, at least one of <publisher> or <publisher_office> shall be specified as well. Its syntax is given in Relax NG compact format below and explained in the following text.

```
publisher_info =
  element publisher_info {
    attlist_publisher_info,
    ((publisher, publisher_office) | publisher | publisher_office),
    publisher_other_info?
  }
attlist_publisher_info &= empty
```

[Child elements]

<publisher> Stores the information about the publisher if it is an individual. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
publisher =
  element publisher { attlist_publisher, publisher_name, address_info? }
attlist_publisher &= empty
```

[Child elements]

<publisher_name> The name of the publisher is recorded as an external character string in this element. Shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
publisher_name =
  element publisher_name {attlist_publisher_name, TextWithGaiji }
attlist_publisher_name &= attribute reading { text }?
```

[Attribute]

reading: gives the pronunciation of the publisher's name as a reading. May be omitted.

<address_info> Stores the address of the publisher, in the standard address format. May be omitted. The details of the <address_info> element are given in A.3.22.

<publisher_office> Stores the information about the publisher if it is a company. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
publisher_office =
  element publisher_office {
    attlist_publisher_office, organization_name, address_info?
  }
attlist_publisher_office &= attribute publisher_code { text }?
```

[Attribute]

publisher_code: Records the publisher's ID. May be omitted.

[Child elements]

<organization_name> The organization's name is recorded in the element. May be omitted. The details of the <organization_name> element are given in A.3.21.

<address_info> The publisher's address is recorded in this element, in the standard address format. May be omitted. The details of the <address_info> element are given in A.3.22.

<publisher_other_info> Other information related to the publisher may be stored as an external character string in this element. May be omitted. If the information is to be displayed, the space character (U+0020), and the linefeed and carriage return characters (U+000D, U+000A, U+000D+U+000A) are to be displayed as is, but the tabulation character (U+0009) is to be displayed as if it were a space. Its syntax is given in Relax NG compact format below.

```
publisher_other_info =
  element publisher_other_info {
    attlist_publisher_other_info, TextWithGaiji
  }
attlist_publisher_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
```

<seller_info> Stores the information related to the seller. May be omitted. If <seller_info> is specified, at least one of <seller> or <seller_office> shall be specified as well. Its syntax is given in Relax NG compact format below and explained in the following text.

```
seller_info =
  element seller_info {
    attlist_seller_info,
    ((seller, seller_office) | seller | seller_office),
    seller_other_info?
  }
attlist_seller_info &= empty
```

[Child elements]

<seller> Stores the information about the seller if it is an individual. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
seller = element seller { attlist_seller, seller_name, address_info? }
attlist_seller &= empty
```

[Child elements]

<seller_name> The name of the seller is recorded as an external character string in this element. Shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
seller_name = element seller_name { attlist_seller_name,
                TextWithGaiji }
attlist_seller_name &= attribute reading { text }?
```

[Attribute]

reading: gives the pronunciation of the seller's name as a reading. May be omitted.

<address_info> Stores the address of the seller, in the standard Address format. May be omitted. The details of the <address_info> element are given in A.3.22.

<seller_office> Stores the information about the seller if it is a company. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
seller_office =
  element seller_office {
    attlist_seller_office, organization_name, address_info?
  }
attlist_seller_office &= attribute seller_code { text }?
```

[Attribute]

seller_code: Records the seller's ID. May be omitted.

[Child elements]

<organization_name> Records the organization's name. May be omitted. The details of the <organization_name> element is given in A.3.21.

<address_info> Records the seller's address, written in the standard Address format. May be omitted. The details of the <address_info> element are given in A.3.22.

<seller_other_info> Other information related to the seller may be stored as an External character string in this element. May be omitted. If the information is to be displayed, the space character (U+0020), and the linefeed and carriage return characters (U+000D, U+000A, U+000D+U+000A) are to be displayed as is, but the tabulation character (U+0009) is to be displayed as if it were a space. Its syntax is given in Relax NG compact format below.

```
seller_other_info =
  element seller_other_info { attlist_seller_other_info, TextWithGaiji }
attlist_seller_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
```

<book_id_info> Records the book's identification number, such as its ISBN number. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
book_id_info = element book_id_info { attlist_book_id_info, book_id+ }
attlist_book_id_info &= empty
```

[Child elements]

<book_id> Each type of identification number is stored in a <book_id> element, written as a standard character string. When <book_id_info> is not omitted, there shall be at least one <book_id>. Its syntax is given in Relax NG compact format below and explained in the following text.

```
book_id = element book_id { attlist_book_id, text }
attlist_book_id &= attribute type { text }
```

[Attribute]

type: Specifies the type of the identification number, such as "ISBN" for instance. Written as a standard character string. Shall not be omitted.

<classification_info> Stores information on the classification of the book. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
classification_info =
  element classification_info {
    attlist_classification_info, classification+
  }
attlist_classification_info &= empty
```

[Child elements]

<classification> Each different type of classification is stored in a separate <classification> element. When <classification_info> is not omitted, there shall be at least one <classification>. It is stored as an external character string. Its syntax is given in Relax NG compact format below and explained in the following text.

```
classification =
  element classification { attlist_classification, TextWithGaiji }
attlist_classification &= attribute type { text }
```

[Attribute]

type: type of the classification used. Shall not be omitted.

<rating> Allows to rate the contents as violent, or adult. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
rating = element rating { attlist_rating, empty }

attlist_rating &=

 [ a:defaultValue = "no" ] attribute adult { Yes_No }?,

 [ a:defaultValue = "no" ] attribute violence { Yes_No }?
```

[Attributes]

adult: Rates the contents as adult oriented materials. Possible values are "yes" or "no". Defaults to no in case of omission.

violence: Rates the contents as violent. Possible values are "yes" or "no". Defaults to no in case of omission.

<publication_place> The country of publication is recorded as a standard country in this element. May be omitted.

<publication_date_info> Stores information regarding the publication date of the book. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text. Each relevant date is to be stored in separate instances of the following child element.

```
publication_date_info =
  element publication_date_info {
    attlist_publication_date_info, publication_date+
  }
attlist_publication_date_info &= empty
```

[Child elements]

<publication_date> Stores a date relevant to the publication, such as the publication date itself as well as other dates such as the printing date, or the beginning of sales date, etc. Each date is stored in the standard Date format. If <publication_date_info> is not omitted, there shall be at least one <publication_date>. Its syntax is given in Relax NG compact format below and explained in the following text.

```
publication_date =
  element publication_date { attlist_publication_date, text }
attlist_publication_date &=
  [ a:defaultValue = "publish" ] attribute type { "publish" | "sale" }?
```

[Attribute]

type: Specifies what type of date it is. "publish" if it is the publication date and "sale" if it is the beginning of sales date. If omitted, it will default to "publish".

<net_price_info> Defines the price of the book. May be omitted, if the price is open, or not set. Its syntax is given in Relax NG compact format below and explained in the following text. More than one price, classified by currency and country, may be stored in separate instances of the following child element.

```
net_price_info =
  element net_price_info { attlist_net_price_info, net_price+ }
attlist_net_price_info &= empty
```

[Child elements]

<net_price> Stores a price specific to one country and currency, written as a standard character string. If <net_price_info> is not omitted, there shall be at least one <net_price>. Its syntax is given in Relax NG compact format below and explained in the following text.

```
net_price = element net_price { attlist_net_price, text }
attlist_net_price &=
  attribute country { text }?,
  attribute unit { text },
  attribute other_info { text }?
```

[Attributes]

country: Defines the country in which this price should apply, in the standard country data format. If omitted, it applies to all countries.

unit: Defines the currency, as a standard character string. Shall not be omitted.

other_info: Other information, written as a standard character string. May be omitted.

<book_abstract> An abstract of the book, written as an external character string, is recorded in this element. May be omitted. If the information is to be displayed, the space character (U+0020), and the linefeed and carriage return characters (U+000D, U+000A, U+000D+U+000A) are to be displayed as is, but the tabulation character (U+0009) is to be displayed as if it were a space. Its syntax is given in Relax NG compact format below.

```
book_abstract =
  element book_abstract { attlist_book_abstract, TextWithGaiji }
attlist_book_abstract &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
```

<front_cover_image> Defines the image to use as front cover image, by recording its location as a standard filename. Currently, JPEG and PNG images are supported. May be omitted. The file type as defined by it shall be checked against the type attribute before opening the file. Its syntax is given in Relax NG compact format below and explained in the following text.

```
front_cover_image =
  element front_cover_image { attlist_front_cover_image, text }
attlist_front_cover_image &= attribute type { text }
```

[Attribute]

type: Defines the type of the image, by giving its MIME type, for example "image/jpeg". Shall not be omitted.

<spine_cover_image> Define the image to use as a spine image. Follows the same rules as <front_cover_image>. May be omitted. Its syntax is given in Relax NG compact format below.

```
spine_cover_image =
  element spine_cover_image { attlist_spine_cover_image, text }
attlist_spine_cover_image &= attribute type { text }
```

<keyword_list> Records a list of keywords related to the book's data, written in the keyword syntax. May be omitted. Its syntax is given in Relax NG compact format below.

```
keyword_list = element keyword_list { attlist_keyword_list, keyword+ }
attlist_keyword_list &= empty
```

<other_book_info> Other information related to the book may be stored as an external character string in this element. May be omitted. If the information is to be displayed, the space character (U+0020), and the linefeed and carriage return characters (U+000D, U+000A, U+000D+U+000A) are to be displayed as is, but the tabulation character (U+0009) is to be displayed as if it were a space. Its syntax is given in Relax NG compact format below.

```
other_book_info =
  element other_book_info { attlist_other_book_info, TextWithGaiji }
attlist_other_book_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
```

An example of Bibliographical data is shown in Figure A.22.

```
<book_info>
	<title_info>
		<series_title>Dummy books</series_title>
		<title>the dummy book of nonsense</title>
		<edition_info>2000/01/01 first edition,
				2005/01/01 second edition</edition_info>
	</title_info>
	<author_info>
		<author role="author">
			<personal_name>
				<first_name>John</first_name>
				<last_name>Smith</last_name>
			</personal_name>
			<address_info>
				<mail_address>john.smith@abcd.com</mail_address>
				<website>http://www.abcd.com/~jsmith/</website>
			</address_info>
		</author>
	</author_info>
	<publisher_info>
		<publisher_office>
			<organization_name>abcd corporation</organization_name>
			<address_info>
				<postal_code>100-1000</postal_code>
				<address>1 main street, Foobar city, Japan</address>
			</address_info>
		</publisher_office>
	</publisher_info>
	<book_id_info>
		<book_id type="ISBN">xxx-x-xxxx-xxxx-x</book_id>
		<book_id type="Japanese_ID_number">454745-7</book_id>
	</book_id_info>
	<classification_info>
		<classification type="Japanese_C_CODE">2143</classification>
	</classification_info>
	<rating adult="no" violence="no"/>
	<publication_place>jpn</publication_place>
	<publication_date_info>
		<publication_date type="publish">2005</publication_date>
	</publication_date_info>
	<net_price_info>
		<net_price country=”jpn” unit="yen">1200</net_price>
	</net_price_info>
	<book_abstract>This book doesn’t talk about anything special.</book_abstract>
	<front_cover_image type=”image/png”>xxx.png</front_cover_image>
	<spine_cover_image type=”image/png”>yyy.png</spine_cover_image>
	<keyword_list>
		<keyword>dummy</keyword>
		<keyword>nonesense</keyword>
	</keyword_list>
</book_info>
```


Figure A.22 – Examples of the book_info element with its descendants

A.4.3 Content management module <body_module>

A.4.3.1 General

The content management module (<body_module>) is in charge of coordinating the contents data into making the actual document. Its syntax is given in Relax NG compact format below and explained in the following text.

```
body_module =
  element body_module { attlist_body_module, flow_type_body }
attlist_body_module &= empty
[Attribute]
None.
```

[Child elements]

<flow_type_body> Handles the flowing contents' data. See A.4.3.2 for details.

An example is shown in Figure A.23.

```
<body_module>
        <flow_type_body>
                <flow_entry>
                        <flow_data ...> ... </flow_data>
                        <flow_data ...> ... </flow_data>
                        ...
                </flow_entry>
                <special_page_link>
                        <special_page kind="contents">PG0001</special_page>
                        <special_page kind="body">PG0002</special_page>
                </special_page_link>
                <search_table>
                ...
                </search_table>
        </flow_type_body>
</body_module>
```

Figure A.23 – Example of the body_module element with its descendants

A.4.3.2 Flowing content data <flow_type_body>

A.4.3.2.1 <flow_type_body> element

The <flow_type_body> element handles the flowing contents' data. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flow_type_body =
  element flow_type_body {
    attlist_flow_type_body,
    flow_entry,
    special_page_link?,
    search_table?
  }
```

```
attlist_flow_type_body &= empty
```

[Attribute]

None.

[Child elements]

<flow_entry>	Registers the flow data to be used as the main text's flowing content. There shall only be one instance of this element. Shall not be omitted. See A.4.3.2.2 for details.
<special_page_link>	Special page data. Allows specifying the position in the flowing content of some often needed pages, such as the index, or the beginning of the main content, for easy reference. May be omitted. Written as described in A.4.3.2.5. Omission of this element means there is no special page.
<search_table>	Records the data needed to create a search table. May be omitted. Written as described in A.4.3.2.6. Omission of this element means there is no search table.

A.4.3.2.2 Flow data registering module <flow_entry>

The <flow_entry> element registers the flow data to be used as the main text's flowing content. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flow_entry =
  element flow_entry {
    attlist_flow_entry, flow_default_attribute?, flow_data+
  }
attlist_flow_entry &= empty
```

[Child elements]

<flow_default_attribute>	Sets the default attributes to be used to display each flow data, as defined by the following <flow_data> element. May be omitted. See A.4.3.2.3 for details. When omitted, the viewer should behave as if all of its attributes and child elements were set to their default value.
<flow_data>	Registers information on each flow data. There is a one to one relation between the number of <flow_data> elements and actual flow data to be recorded/displayed. There shall be one or more instances of this element. The order in which flow data are recorded determines the display order. See A.4.3.2.4 for details.

A.4.3.2.3 Flow data default attribute module <flow_default_attribute>

Sets the default values of attributes that will be used to display each flow data (see A.4.3.2.4), of which the main flowing content is composed. The values set in this element will serve as default values for all the flows which have a text object, a search screen object or dictionary data object as main data.

Part of the values that may be set in this element may also be set locally in the text object instance of each content data. In order to set a default value for particular content data, the

<text_default_attribute> of the text object instances (see A.4.6.2) or <dict_default_attribute> of the dictionary data object designated by the content data should be used. If the value defined by the <flow_default_attribute> element for the whole content data and the value set in an individual content data's <text_default_attribute> or <dict_default_atribute> are in conflict, the latter has priority. When neither is set, the behaviour is not defined by this annex, and depends on the viewer's default, or the user's preferences. Moreover, if the viewer does not implement the required method, or wishes to give priority to user settings, it may proceed without respecting the value defined in these default value elements, except when the following explanations state otherwise.

The syntax of the <flow_default_attribute> element is given in Relax NG compact format below and explained in the following text. If it is omitted, the viewer should behave as if all of its attributes and child elements were set to their own default value.

```
flow_default_attribute =
  element flow_default_attribute {
    attlist_flow_default_attribute,
    flow_default_size?,
    flow_default_font?,
    flow_default_background?,
    flow_default_line_breaking_method?
  }
attlist_flow_default_attribute &=
  attribute baseline { BaseLine }?,
  attribute view_type { ViewType }?,
  [ a:defaultValue = "IPA" ] attribute phonetic_notation { text }?
```

[Attributes]

baseline: Defines the orientation of the baseline (and therefore of the text) for each flow. The possible values are listed below. May be omitted. If neither this attribute nor the corresponding option in each content data (the baseline attribute of the <text_default_attribute> element in text object instance or <dict_default_attribute> in dictionary data object instance, see A.4.6.2.2 and A.4.6.3.2 for details) is set, the default value depends on the viewer.

- "right": The writing direction is horizontal (left to right). However, the direction can be changed at the user's option.
- "right_only": The writing direction is horizontal (left to right), and the user cannot change the setting. However, if it is not supported by the viewer, this setting is not necessarily applied.
- "down": The writing direction is vertical (top to bottom). However, the direction can be changed at the user's option.
- "down_only": The writing direction is vertical (top to bottom), and the user cannot change the setting. However, if it is not supported by the viewer, this setting is not necessarily applied.

view_type: Defines the default screen orientation for each flow. The possible values are listed below. May be omitted. When omitted, the default value depends on the viewer.

- "portrait": Chooses portrait (taller than wide) mode. However, it can be set to another direction at the user's option.

"portrait_only": Chooses portrait (taller than wide) mode, and the user cannot change the setting. However, if the viewer cannot handle this screen orientation, this is not necessarily applied.

"landscape": Chooses landscape (wider than tall) mode. However, it can be set to another direction at the user's option.

"landscape_only": Chooses landscape (wider than tall) mode, and the user cannot change the setting. However, if the viewer cannot handle this screen orientation, this is not necessarily applied.

phonetic_notation: Defines the default pronunciation notation of the content. Defaults to "IPA". This is overridden by the phonetic notation attribute of <pronunciation> and <headword> elements inside such elements.

An example is shown in Figure A.24.

```
<flow_default_attribute  base_line="right"  view_type="portrait>
```

Figure A.24 – Example of the flow_default_attribute element

[Child elements]

<flow_default_size> Defines the default letter spacing, line pitch and margin size for all the flows of the content data. May be omitted. If it is, the viewer should behave as if all of its attributes were set to their default value. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flow_default_size =
  element flow_default_size { attlist_flow_default_size, empty }
attlist_flow_default_size &=
  attribute letter_spacing { FiveSize }?,
  attribute line_pitch { FiveSize }?,
  attribute margin { "big" | "medium" | "small" }?
```

[Attributes]

letter_spacing: Default letter spacing. The following values are allowed. May be omitted. If it is, the size is unspecified and depends on the viewer.

"maximum"

"big"

"medium"

"small"

"minimum"

The actual sizes of these 5 possibilities are viewer-dependent, as it depends on the capabilities of the underlying device.

line_pitch: Default line pitch. The following values are allowed. May be omitted. If it is, the size is unspecified and depends on the viewer.

"maximum"

"big"

"medium"

"small"

"minimum"

The actual sizes of these 5 possibilities are viewer-dependent, as they depend on the capabilities of the underlying device.

margin: Default margin size. The following values are allowed. May be omitted. If it is, the size is unspecified and depends on the viewer.

"big"

"medium"

"small"

The actual sizes of these 3 possibilities are viewer-dependent, as they depend on the capabilities of the underlying device.

<flow_default_font> Defines the default font name, size, and properties for all the flows of the content data. May be omitted. If an individual flow of the content data also defines it (with the <text_default_font> element of the text object instance as defined in A.4.6.2.2 and <dict_default_font> element of the dictionary data object instance as defined in A.4.6.3.2, the individual values take precedence. If both are omitted, the behaviour should correspond to the default value of the attributes listed below. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flow_default_font =
  element flow_default_font { attlist_flow_default_font, empty }
attlist_flow_default_font &=
  attribute fontname { text }?,
  attribute fontsize { FiveSize }?,
  attribute bold_flag { Yes_No }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute ruby_flag { "yes" | "yes_only" | "no" | "no_only" }?,
  attribute italic { Yes_No }?,
  attribute oblique { Yes_No }?,
  attribute small_caps { Yes_No }?,
  attribute family {
    "monospace" | "san-serif" | "serif" | "cursive" | "fantasy"
  }?,
```

```
attribute ul_type { "rightscore" | "leftscore" | "throughscore" }?,
attribute em_type {
  "rightscore"
  | "leftscore"
  | "throughscore"
  | "kendot"
  | "bold"
  | "italic"
  | "bold_italic"
  | "reverse"
  | "shade"
}?
```

[Attributes]

fontname: Default font name. More than one font may be specified. In that case, each font name should be separated by a comma (U+002C). For instance:

fontname="Aaa sans serif,Bbb gothic"

The viewer should use the first listed font that is available. May be omitted. If both this attribute and the individual flow's attribute are omitted, the default value depends on the viewer.

fontsize: Default font size. The following values are allowed. May be omitted. If it is, the size is unspecified and depends on the viewer.

"maximum"

"big"

"medium"

"small"

"minimum"

The actual letter sizes corresponding to these 5 possibilities are viewer-dependent, as it depends on the capabilities of the underlying device.

bold_flag: Specifies if the content data should be displayed in bold or not. If omitted, the behaviour depends on the viewer. The acceptable values are:

"yes": display as bold

"no": display normally

If set to "yes", all characters shall be displayed in bold style, except, in a text object instance or a dictionary data object instance, those within a <font> element with its bold_flag attribute set to "no" (see A.4.6.2.3 and A.4.6.3.4).

color_space, color, opacity:

Defines the font colour to be used for the content data. Written in the standard colour data type. If omitted, the value depends on the viewer.

ruby_flag: Defines whether ruby in the content data is to be viewed or not. When omitted, behaviour depends on the viewer. The

text to be displayed when ruby is turned on is the one included in the <ruby> element of text object instances or dictionary data object instances. The following values may be used.

"yes": Ruby should be displayed, but can be turned off by the user.

"yes_only": Ruby shall be displayed, and cannot be turned off by the user. However, this does not apply to viewers not able to display ruby.

"no": It is recommended that ruby should not be displayed, but can still be turned on by the user.

"no_only": Ruby shall not be displayed and cannot be turned on by the user. However, this does not apply if the viewer is incapable

italic: Decides if the content data should be displayed in italic or not. If omitted, the behaviour depends on the viewer. The acceptable values are:

"yes": display in italic

"no": display normally

If set to "yes", all characters shall be displayed in italic style, except, in a text object instance or a dictionary data object instance, those within a <font> element with its italic attribute set to "no" (see A.4.6.2.3 and A.4.6.3.4).

oblique: Decides if the content data should be displayed in oblique or not. If omitted, the behaviour depends on the viewer. The acceptable values are

"yes": display in oblique;

"no": display normally.

If set to "yes", all characters shall be displayed in oblique style, except, in a text object instance or a dictionary data object instance, those within a <font> element with its oblique attribute set to "no" (see A.4.6.2.3 and A.4.6.3.4).

small_caps: Decides if the content data should be displayed in small capitals or not. If omitted, the behaviour depends on the viewer. The acceptable values are

"yes": display in small capitals;

"no": display normally.

If set to "yes", all characters shall be displayed in small capitals, except, in a text object instance or a dictionary data object instance, those within a <font> element with its small_caps attribute set to "no" (see A.4.6.2.3 and A.4.6.3.4).

family: Specifies font family to be used. The font actually used in rendering depends on the viewer. If omitted, the behaviour is also viewer-dependent. The acceptable values are

"monospace": display in fixed-width fonts;

"sans-serif": display in fonts without serifs;

"serif" display in fonts with serifs;

"cursive" display in handwriting-style fonts.

"fantasy" Specifies in decorative fonts. If set to "yes", all characters shall be displayed in decorative style, except, in a text object instance or a dictionary data object instance, those within a <font> element with its fantasy_flag attribute set to "no" (see A.4.6.2.3 and A.4.6.3.4).

ul_type Specifies how strings in <u> element and its equivalent (<font underline="yes">) should be rendered. The acceptable values are as follows, all of which are taken from ISO/IEC 9541-1:2012, 8.7.1.12.1.1. Refer to ISO/IEC 9541-1:2012 for the exact meaning of these values.

"rightscore"

"leftscore"

"throughscore"

em_type: Specifies how strings in elements should be rendered. The acceptable values are as follows, the first four of which are taken from ISO/IEC 9541-1:2012, 8.7.1.12.1.1. Refer to ISO/IEC 9541-1:2012 for the exact meaning of these values.

Note that when this attribute is specified, element is no longer equivalent to <font bold="yes" italic="yes">.

"rightscore"

"leftscore"

"throughscore"

"kendot"

"bold": The substrings should be rendered in bold font.

"italic": The substrings should be rendered in italic font.

"bold_italic" The substrings should be rendered in bold italic font.

"reverse" The substring should be rendered with the characters and background reversed.

"shade" The substrings should be rendered with shade.

<flow_default_background> Defines the background colour to be used for all the flows of the content data. If an individual flow of the content data also defines it (with the <text_default_font> element of the text object instance, as defined in A.4.6.2.2 or with the <dict_default_font> element of the dictionary data object instance as defined in A.4.6.3.2, the individual values take precedence. If both are omitted, the behaviour should correspond to the default value of the attributes listed below. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flow_default_background =
  element flow_default_background {
    attlist_flow_default_background, empty
  }
attlist_flow_default_background &=
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?
```

[Attributes]

color_space, color, opacity:

Defines the font colour to be used for the content data. Written in the standard colour data type. If omitted, the value depends on the viewer.

<flow_default_line_breaking_method> Specifies the algorithm to be used to determine how text should be split in lines. Various languages handle this in various ways, so this element allows for some flexibility. May be omitted. If both this element and <line_breaking_method> elements in the text/dictionary data object instance (defined in A.4.6.2.2 and A.4.6.3.2 are specified, the latter takes priority. If both are omitted, the default values of this <flow_default_line_breaking_method> element define the behaviour. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flow_default_line_breaking_method =
  element flow_default_line_breaking_method {
    attlist_flow_default_line_breaking_method, empty
  }
attlist_flow_default_line_breaking_method &=
  [ a:defaultValue = "none" ] attribute method { "none" }?
```

[Attributes]

method: Chooses the line breaking method. Defaults to "none". Possible values are:

"none": no special processing. When a line is filled with characters, go to the next one.

Other values are possible for localization. See Clause A.6 for details.

Other attributes may be added by localized methods. See Clause A.6for details.

[Child elements]

None, unless added by localized methods. See Clause A.6 for details.

<flow_default_delimiter> Specifies the block for delimiters (character strings that appear before the first character and after the last character) for various elements in the content. Its child elements are elements that specify such delimiters for some elements, which serve as a guide when the content is converted into the data for the viewer, or rendered directly. <flow_default_delimiter> may be omitted, and all the delimiters default to "" (empty string) in such cases. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flow_default_delimiter =
  element flow_default_delimiter {
    attlist_flow_default_delimiter, delimiter?
  }
attlist_flow_default_delimiter &= empty
```

[Child elements]

<delimiter> Defines the starting and ending delimiters of the child element of the element specified by the element attribute.

It is also possible to specify the type attribute when applicable (i.e. the element specified has the type attribute). The element shall be ignored if the element attribute specifies a non-text-rendering element. When no delimiter is specified to a element, it is regarded that an empty string "" is specified. The syntax of the <delimiter> element is given in Relax NG compact format below and explained in the following text.

```
delimiter = element delimiter { attlist_delimiter, empty }
attlist_delimiter &=
  attribute tag { text },
  attribute type { text }?,
  [ a:defaultValue = "" ] attribute open { text }?,
  [ a:defaultValue = "" ] attribute close { text }?
```

[Attributes]

tag: Specifies which element the delimiters apply to. Ignored when a non-existing element is specified. Shall not be omitted.

type: Specifies the value of the type attribute of the element which the delimiters apply to. When type attribute is specified to a element that has no type attribute, the instance of <delimiter> element is ignored. If multiple <delimiter> elements are given, and one of them has the type attribute while the others don't, the former overrides the latter when the former is applicable. May be omitted.

open: Defines the starting (opening) delimiter. May be omitted and defaults to "" (empty string).

close: Defines the ending (closing) delimiter. May be omitted and defaults to "" (empty string).

[Child elements]

None.

An example is shown in Figure A.25.

```
<flow_default_delimiter>
<delimiter tag="headword" type="pronunciation" open="/" close="/>
<delimiter tag="etymology" open="&lt;" close="&gt;"/>
<delimiter tag="p" open="*"/>
</flow_default_delimiter>
```

Figure A.25 – Example of the flow_default_delimiter element with its descendants

A.4.3.2.4 Flow data <flow_data>

This is where flow data is defined. Under this element, the object to be used as the flow's content is registered, as well as other information such as page links and events. The syntax of the <flow_data> element is given in Relax NG compact format below and explained in the following text.

```
flow_data = element flow_data { attlist_flow_data, event_info? }
attlist_flow_data &=
  attribute flow_id { text }?,
  attribute body_id { text },
  [ a:defaultValue = "off" ]
  attribute turning_page_control { Turn_Page_Val }?
```

[Attributes]

flow_id: Sets the ID number of the flow data, written in the form of a Page_ID. May be omitted. Within <flow_entry>, there shall be no other flow data with the same ID.

body_id: Points to the object that will constitute this flow's content, using the Object_ID of a "flowing content text" object, or of a "search page" object, or a dictionary data object. Shall not be omitted.

turning_page_control: The viewer allows moving forward or backward in the contents. However, it is possible to restrict moves to the previous or next flow by setting this attribute to one of the values listed below. When omitted, it defaults to "off". Note, however, that when body id points to a search page object, turning_page_control shall be set to "on".

- "on": Moving both to the next or previous flows is forbidden.
- "off": Moving to the next and previous flows is permitted.
- "forward": Moving to the next flow is forbidden, but moving to the previous one is allowed.
- "back": Moving to the next flow is permitted, but moving to the previous one is forbidden.

As explained above, this setting limits moves from one flow to the others, but they do not restrict moves within each flow. To restrict moves inside a flowing content text object, the turning page control of the <page_break> element (see A.4.6.2.3) shall be used in flowing content text object and of the <dic-item> element in dictionary data object.

[Child elements]

<event_info> Event information module. Defines events (triggers) and associated actions. May be omitted. See A.4.4 for details.

A.4.3.2.5 Special page data <special_page_link>

Records the position of important or frequently accessed pages, to make it easier to jump to these parts of the document. For instance, easy access to the map in a travel guide, to the chronology in a history book, or the glossary in a technical paper can prove useful. The syntax of the <special_page_link> element is given in Relax NG compact format below and explained in the following text.

```
special_page_link =
  element special_page_link { attlist_special_page_link, special_page+ }
attlist_special_page_link &= empty
```

[Child elements]

<special_page> Records the information about a single special page. Its syntax is given in Relax NG compact format below and explained in the following text.

```
special_page = element special_page { attlist_special_page, text }
attlist_special_page &=
  [ a:defaultValue = "other" ]
  attribute kind {
    "cover"
    | "title_page"
    | "preface"
    | "contents"
    | "body"
    | "column"
    | "note"
    | "figure"
    | "ad"
    | "afterword"
    | "appendix"
    | "answer"
    | "glossary"
    | "bibliography"
    | "commentary"
    | "index"
    | "imprint"
    | "author_info"
    | "other"
```

}?,
attribute title { text }?

[Attributes]

kind: Describes the contents of the page referred to. The possible values are listed below. Defaults to "other" when omitted.

"cover", "title_page", "preface", "contents" (table of contents), "body" (beginning of the content), "column" (boxed piece of text), "note", "figure", "ad", "afterword", "appendix", "answer", "glossary", "bibliography", "commentary", "index", "imprint", "author_info", "other".

title: Defines a title for the position of the document referred to as a special page. Written as an extended character string. May be omitted.

[Child elements]

Standard character string: Specifies the position to consider as a special page, by choosing a particular flow data from the whole flowing content. Shall not be omitted. For a textual flow data, it is possible to specify a position within it as well.

– To specify only the flow data as a whole.

Written as a Page_ID. It records the corresponding flow data's page ID.

An example is shown in Figure A.26.

```
<special_page ... >PG0001</special_page>
```

Figure A.26 – Example of the special_page element

– To specify the flow data and the position within it.

Written as "Page_ID/Object_ID/Char_ID". Page ID is the flow data's ID number. Object ID is the ID number of a text/dictionary data object within that flow. Char ID is the ID number of a string defined within that text/dictionary data object.

An example is shown in Figure A.27.

```
<special_page ... >
        PG0001/OB0321/CR0982
</special_page>
```

Figure A.27 – Example of the special_page element

Another example of the special_page_link element with its descendants is shown in Figure A.28.

```
<special_page_link>
        <special_page kind="contents">PG1111 </special_page>
        <special_page kind="other" title="Downtown area map">PG1114/OB0022/CR0001
        </special_page>
        ...
</special_page_link>
```

Figure A.28 – Example of the special_page_link element with its descendants

A.4.3.2.6 Search table data <search_table>

Information defining the search tables is stored here, such as the search table's ID, or parameters concerning entry words registered in the search table. Entry words hereafter means target of each search and represented in two different ways:

- registered in the <head> element of the text object instance (see A.4.6.2.3);
- registered in the <head> element of the dictionary data object instance (see A.4.6.3.4).

During the conversion to the distribution format, based on the parameter stored in the <search_table_def> element defined below, the actual search table is built, after verification of the entry word strings. In the distribution format, entry words are stored in this search table. The syntax of the <search_table> element is given in Relax NG compact format below and explained in the following text.

```
search_table =
  element search_table { attlist_search_table, search_table_def+ }
attlist_search_table &=
  [ a:defaultValue = "no" ] attribute bookmark { Yes_No }?,
  [ a:defaultValue = "no" ] attribute wordbook { Yes_No }?,
  [ a:defaultValue = "no" ] attribute jump_search_root { Yes_No }?,
  [ a:defaultValue = "no" ] attribute jump_search { Yes_No }?,
  [ a:defaultValue = "no" ] attribute all_search { Yes_No }?
```


[Attributes]

bookmark: Specifies if all the search tables included in the content should be eligible for automatic bookmarking on the viewer (i.e. automatically keeping the tally of the headwords searched). The following values are possible. May be omitted and regarded as "no" when omitted.

"yes": the search tables are eligible for automatic bookmarking.

"no": the search tables are not eligible for automatic bookmarking.

wordbook: Specifies if all the search tables are eligible for the wordbook function on the viewer (i.e. automatically registering headwords in the wordbook and showing the list). The following values are possible. May be omitted and regarded as "no" when omitted.

"yes": the search tables are eligible for wordbook function.

"no": the search tables are not eligible for wordbook function.

jump_search_root: Specifies if the content is eligible as a source of multi-content search from text (i.e. can launch multi-content search using the keyword selected from the displayed text of the content). The following values are possible. May be omitted and regarded as "no" when omitted.

"yes": the content is eligible as a source of multi-content search from text.

"no": the content is not eligible as a source of multi-content search from text.

jump_search: Specifies if all the search tables are eligible for multi-content search from text (i.e. can be searched as one of the contents in multi-content search using the keyword selected from the text) and multi-content search from input with the content being displayed (i.e. can be searched as one of the contents in multi-content search using the input keyword) with the content displayed. The following values are possible. May be omitted and regarded as "no" when committed.

"yes": the search tables are eligible for such searches.

"no": the search tables are not eligible for such searches.

all_search: Specifies if all the search tables are eligible for multi-content search without the content itself being displayed. The following values are possible. May be omitted and regarded as "no" when omitted.

"yes": the search tables are eligible for such searches.

"no": the search tables are not eligible for such searches.

If specified in the text object instances, only "no" is valid for bookmark/wordbook/jump_search_root/jump_search/all_search attributes.

[Child elements]

<search_table_def> Specifies information concerning each search table. There shall be at least one <search_table_def> element in the <search_table> element. Its syntax is given in Relax NG compact format below and explained in the following text.

```
search_table_def =
  element search_table_def {
    attlist_search_table_def, enable_key_type, key_normalization
  }
attlist_search_table_def &=
  attribute id { text },
  [ a:defaultValue = "no" ] attribute use_default { Yes_No }?,
  [ a:defaultValue = "unicode" ] attribute sorting_rule { "unicode" }?,
  attribute name { text },
  attribute short_name { text },
  attribute wild { Yes_No },
  attribute blank { Yes_No },
  attribute end { Yes_No },
  attribute help_page_id { text }?
```

[Attributes]

id: Defines the ID number of this search table. The ID number shall be unique to each search table within the content and stored as a standard character string. Shall not be omitted. During the conversion to the distribution format, the search table is built according to the parameters in this <search_table_def> element, and added to the contents. This ID number is used to refer to the search table either from the search page's entry fields (see A.4.6.7 for details) or from the entry word information stored in text object instance's <head> element (see A.4.6.2.3 for details). Not to be confused with Reference ID (see A.4.6.3.4).

use_default: Defines whether a search based on this search table is allowed by the viewer, when the search is not initiated from a related search page. The following values may be used. Defaults to "no" when omitted.

"yes" such search is allowed.

"no" such search is not allowed.

sorting_rule: Defines how to sort the search results. This attribute is to allow for different ordering schemes for different languages. May be omitted. Defaults to "unicode". The following values may be used.

"unicode": The rank of each character is determined by its Unicode code point.

Other sorting methods are defined in the localization (see A.6.5).

name: Sets the name for the search table. Written in a standard character string. The name is intended to be used in listing the search results. Shall not be omitted.

short_name: Sets the shortened version of the name of the table. Names longer than the one specified by the name attribute are not allowed. The name is intended to be used in listing the search results. Shall not be omitted.

wild: Specifies if the search table for wildcard search (i.e. it is known how many letters need to be filled in the blanks of the word as well as the positions of the blanks) should be output. The following values are possible. Shall not be omitted.

"yes" the table for wildcard search should be output.

"no" otherwise.

blank: Specifies if the search table for blank word search (where only the positions of the blanks in the word are known) should be output. The following values are possible. Shall not be omitted.

"yes" the table for blank word search should be output.

"no" otherwise.

end: Specifies if the search table for word ending search should be output. The following values are possible. Shall not be omitted.

"yes" the table for word ending search should be output.

"no" otherwise.

help_page_id: Specifies the page ID for the flow data that are related to the table (e.g. explanation of the table). May be omitted.

[Child elements]

<enable_key_type> Defines the type of characters that may be used to store the lookup key of the entry words in the table. For some languages such as English, the lookup key will simply be the word itself. However, other languages such as Japanese may use the pronunciation of the word instead of its ideographic representation. Shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
enable_key_type =
  element enable_key_type { attlist_enable_key_type, empty }
attlist_enable_key_type &=
  [ a:defaultValue = "no" ] attribute numerals { Yes_No }?,
  [ a:defaultValue = "no" ] attribute basic_alphabet { Yes_No }?
```

[Attributes]

numerals: Defines whether the lookup key may include numerals or not. Possible values are "yes" and "no". Defaults to "no" when omitted.

basic_alphabet: Defines whether the lookup key may include alphabet (limited to the non-accentuated characters found in US-ACSII) or not. Possible values are "yes" and "no". Defaults to "no" when omitted.

Refer to A.6.6 on localization for language-specific attributes. The character sets above are defined in Table A.5.

Table A.5 – Characters usable for the lookup key

Character set name	Corresponding characters (all values are in Unicode)
numerals	numerals: 0 to 9 (U+0030 to U+0039)
basic_alphabet	Standard US-ACSII alphabet: A to Z (U+0041 to U+005A) and a to z (U+0061 to U+007A)

<key_normalization> Defines the normalization methods to be applied on the keys registered in this search table. Shall not be omitted. The interpretation of this element differs depending on the modes of the search, namely "matches-only" search and "matches-first" search. The mode of search to use is determined by the search_type attribute of the <key_input_region> element (see A.4.6.7 for details).

a) "matches-only" search

(Only the entries matching the input are displayed, being narrowed down as input is becoming complete.) Both user input and the keys registered in the search table are normalized according to the rules set by the following attribute, and matching is conducted on the normalized form.

b) "matches-first" search

(Entries are displayed starting from those matching the input.) The settings are ignored and the normalization is carried out according to the default values of the following attributes.

The syntax of the <key_normalization> element is given in Relax NG compact format below and explained in the following text.

```
key_normalization =
  element key_normalization { attlist_key_normalization, empty }
attlist_key_normalization &=
  [ a:defaultValue = "yes" ] attribute capitalization { Yes_No }?
```

[Attribute]

capitalization: Turn all (alphabetical) characters to upper case. Possible values are "yes" and "no". Defaults to "yes".

See A.6.7 for language-specific normalizations.

An example is shown in Figure A.29.

```
<search_table>
        <search_table_def id="ST0001" use_default="yes">
                <enable_key_type numerals="no" basic_alphabet="yes"/>
                <key_normalization capitalization="yes"/>
        </search_table_def>
        <search_table_def id="ST0002">
        ...
        </search_table_def>
</search_table>
```

Figure A.29 – Example of the search_table element with its descendants

A.4.4 Event info module <event_info>

A.4.4.1 Events

In this annex, audio playback to react to clicks, pages links, or other user activated functions, are called events. The <event_info> element records the events of flow data. Its syntax is given in Relax NG compact format below and explained in the following text.

```
event_info = element event_info { attlist_event_info, event+ }
attlist_event_info &=
  [ a:defaultValue = "single" ] attribute display_type { "single" }?
```

[Attribute]

display_type: Specifies environments for the events. Currently only "single" is possible. Defaults to "single" when omitted.

[Child elements]

<event> Event data. Records information concerning the events that occur after the object pointed by the body_id attribute of the <flow_data> element has been displayed, such as user launched events. For instance, clicking on a string may initiate a jump to another page. See A.4.4.2 for details. There may be more than one instance of this <event> element within <event_info>. If no event is to be specified, the <event_info> element itself shall be omitted.

An example is shown in Figure A.30.

```
<event_info display_type="single">
  <event>
      <trigger>
          <trigger_pointer id="OB0001/CR0001" action_flag="click"/>
      </trigger>
      <action>
          <action_page_jump page_id="PG0002"/>
      </action>
  </event>                                                    Event 1
  <event>
      <trigger>
          <trigger_pointer id=" OB0001/CR0002" action_flag="click"/>
      </trigger>
      <action>
          <action_page_jump page_id="PG0003"/>
      </action>
  </event>                                                    Event2
    ...
</event_info>
```

Figure A.30 – Example of event_info element with its descendants

A.4.4.2 Event data <event>

An event is composed of two parts: the trigger part, and the action part. The former is the condition that triggers the event while the latter describes what is to be done. For example, the trigger may be "the user clicks on a specific area", or "a button is pressed", and the action may be "jump to a specific page", or "play a given soundfile". The syntax of the <event> element is given in Relax NG compact format below and explained in the following text.

```
event = element event { attlist_event, trigger, action }
attlist_event &= empty
```

[Child elements]

<trigger> See A.4.4.3 for details. There shall be only one <trigger> element, and it can have only one child element. Shall not be omitted.

<action> See A.4.4.4 for details. There shall be only one <action> element, and it can have only one child element. Shall not be omitted.

An example is shown in Figure A.31.

```
<event>
    <trigger>
        <trigger_pointer id=" OB0001/CR0002" action_flag="click"/>
    </trigger>                                                    Trigger
    <action>
        <action_page_jump page_id="PG0003"/>
    </action>                                                     Action
</event_info>
```

Figure A.31 – Example of the event element with its descendants

A.4.4.3 Trigger (pointer) <trigger_pointer>

The syntax of the <trigger> element is given in Relax NG compact format below and explained in the following text.

```
trigger = element trigger { attlist_trigger, Trigger_List }
attlist_trigger &= empty
```

[Child elements]

<trigger_pointer> Used to define triggers as click on an area. Its syntax is given in Relax NG compact format below and explained in the following text.

```
trigger_pointer =
  element trigger_pointer { attlist_trigger_pointer, pointer_region? }
attlist_trigger_pointer &=
  attribute id { text },
  [ a:defaultValue = "click" ] attribute action_flag { "click" }?
```

[Attributes]

id: points at the target of the trigger. Shall not be omitted. Written as Object_ID/Char_ID. Not to be confused with Reference ID (see A.4.6.3.4).

An example is shown in Figure A.32.

```
id="OB003k/CR0023"
```

Figure A.32 – Example of the id attribute

The char ID part of this string may only refer to the following (see A.4.6.2.3 for details):

a) the id number defined in the char_id attribute of a <char_id> element;

b) the id number defined in the char_id attribute of an <object> element.

However, when a clickable image map is defined using the <pointer_region> element, only the second case (char_id of an <object> element) may be used.

Note that the id number set in the trigger attribute of a <mask> element (see A.4.6.2.3) shall not be used here. If used, the trigger attribute of the <mask> element is ignored.

action_flag: The type of action which switches the trigger on. In the current document, only "click" is allowed, and is used as a default value in case of omission.

"click": a click in the target zone.

[Child elements]

<pointer_region> When the trigger area is restricted to only a part of the image pointed to by the id attribute of the <trigger_pointer> element, this <pointer_region> element is used to describe a polygonal_region (see A.3.15). If what the id attribute of the <trigger_pointer> points to is not an image, the content of this element is ignored. May be omitted. Its syntax is given in Relax NG compact format below.

```
pointer_region =
  element pointer_region { attlist_pointer_region, vertex+ }
attlist_pointer_region &= empty
```

When events have overlapping triggers, there can be some ambiguity as to which event takes precedence when the overlapping areas are clicked, depending on what kind of pointing devices are used. In that case, events are given precedence in the order of appearance inside the <event_info>. The example shown in Figure A.33 illustrates this situation.

```
<event_info>
        <event>
                <trigger>
                        <trigger_pointer id="OB003k/CR0001" action_flag="click">
                                <pointer_region>
                                        <vertex position="(0,0)"/>
                                        <vertex position="(100,0)"/>
                                        <vertex position="(100,100)"/>
                                        <vertex position="(0,100)"/>
                                </pointer_region>
                        </trigger_pointer>
                </trigger>
                <action>
                        <action_page_jump page_id="PG0043"/>
                </action>
        </event>
        <event>
                <trigger>
                        <trigger_pointer id="OB003k/CR0001" action_flag="click">
                                <pointer_region>
                                        <vertex position="(50,50)"/>
                                        <vertex position="(100,50)"/>
                                        <vertex position="(100,100)"/>
                                        <vertex position="(50,100)"/>
                                </pointer_region>
                        </trigger_pointer>
                </trigger>
                <action>
                        <action_page_jump page_id="PG0021"/>
                </action>
        </event>
</event_info>
```

Figure A.33 – Example of the event_info element with its descendants

In that case, both events are triggered by a click on a sub-area of OB003k/CR0001. These two areas overlap on the (50,50)-(100,100) square. A click in this particular area shall trigger the first of the two events, resulting in a page jump to PG0043.

A.4.4.4 Action information

A.4.4.4.1 <action> element

The syntax of the <action> element is given in Relax NG compact format below. It can only have one instance of its child element.

```
action = element action { attlist_action, Action_List }
attlist_action &=
  [ a:defaultValue = "sequential" ]
  attribute action_flag { "sequential" }?
```

A.4.4.4.2 Playback action <action_play>

Launches the playback of a sound, and animation or other playable items. Its syntax is given in Relax NG compact format below and explained in the following text.

```
action_play = element action_play { attlist_action_play, empty }
attlist_action_play &=
  attribute object_id { text },
  [ a:defaultValue = "normal" ] attribute action { "normal" }?
```

[Attributes]

object_id: Points to the object to be played. Written as described below. Shall not be omitted.

If the object is registered in the object management table (see A.4.5) (in this document, only possible for sound and movie objects), then it is:

Written as an Object_ID.

An example is shown in Figure A.34.

```
object_id="OB003k"
```

Figure A.34 – The object_id attribute

If the object is inserted in the text/dictionary data object instance by the <object> element (in this document, only possible for animations), then it is:

Written as Object_ID/Char_ID.

An example is shown in Figure A.35.

```
object_id="OB003k/CR0023"
```

Figure A.35 – The object_id attribute

Note that the char ID part shall be an id number registered in the char_id attribute of an <object> element (A.4.6.2.3).

action: Defines the playback method. The only accepted value is "normal", and it defaults to "normal" in the case of omission. When the object to be played is a sound item, when the reader moves to another flow, or reaches a <page_break>, or </word> with page_break attribute set to "yes" (in dictionary data object instances only), the playback shall be stopped. In the case of an animation, the playback stops when it goes out of the screen.

Examples are shown in Figure A.36.

```
<action_play object_id="OBkj23"/>
<action_play object_id="OB1234/CR0001"/>
<action_play object_id="OB1234/CR0001" action="normal"/>
```

Figure A.36 – Examples of the action_play element

A.4.4.4.3 Page jump action <action_page_jump>

This elements defines a jump form the current page to another one, or to a website. Its syntax is given in Relax NG compact format below and explained in the following text.

```
action_page_jump =
  element action_page_jump { attlist_action_page_jump, empty }
attlist_action_page_jump &=
  attribute book { text }?,
  attribute book_type { text }?,
  attribute page_id { text }?,
  attribute center { text }?
```

[Attributes]

book: Defines the target document of the jump. If it is the same document as the origin, this attribute should be omitted. Otherwise, there are 3 different usages as listed below.

a) Jump to a website:

   When the target is an html website, the book attribute should be written as a URL address beginning with http:// or https://, as defined in IETF RFC 3986.

b) Make a phone call:

   The book attribute should be written as "tel: " followed by a telephone number.

c) Write a mail to the specified address:

   The book attribute should be written as "mailto: " followed by a single email address.

book_type: Used to differentiate the types of content stored in the book attribute. Shall be omitted if the book is omitted too, shall not be omitted when the book is not omitted.

a) When the book attribute points to a website.

   The attribute shall contain: "text/html"

b) When the book attribute points to a telephone number.

   The attribute shall contain: "application/x-tel"

c) When the book attribute points to a mail address.

   The attribute shall contain: "application/x-mail"

page_id: Defines the target flow data of the jump, when the jump target and the jump origin are within the same document, and shall not be omitted in that case. Written as a Page_ID. If the book attribute is set to a) website, b) telephone, or c) mail address, then this attribute is ignored.

center: Sets a more precise destination to the jump. Can only be used when the jump target is a text/dictionary data object (see A.4.6.2 and A.4.6.3). In that case, the position within that text is specified using the char_id of the target. May be omitted. When omitted, the jump target is the beginning of the flow data set by the page_id attribute. If the book attribute is set to a) website, b) telephone, or c) mail address, then this attribute is ignored.

Examples are shown in Figure A.37.

```
<action_page_jump page_id="PG23k4" />
<action_page_jump page_id="PG23k4" center="CR0001"/>
<action_page_jump book="http://www.sharp.co.jp/" book_type="text/html"/>
```

Figure A.37 – Examples of the action_page_jump element

A.4.5 Parts data module <parts_module>

A.4.5.1 Storage and management

Subclause A.4.5.1 is used to store and manage information about the parts that are used to constitute the flow data. The <object_table> element sets an object ID number and various attributes for all the objects that will be used as content, such as text objects.

The syntax of the <parts_module> element is given in Relax NG compact format below and explained in the following text.

```
parts_module =
  element parts_module { attlist_parts_module, object_table }
attlist_parts_module &= empty
```

[Child elements]

<object_table> Object management module. All objects that are used in the book are registered here. Each object is recorded using one of the elements described in the following subclauses. Shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
object_table =
  element object_table {
    attlist_object_table,
    (dynamic_text_object_entry
     | sound_object_entry
     | search_page_object_entry
     | movie_object_entry
     | dict_data_object_entry)+
  }
attlist_object_table &= empty
```

[Child elements]

The details of these elements are given in A.4.5.2 to A.4.5.6.

<dynamic_text_object_entry> Records text objects serving as a base for the flowing contents.

<sound_object_entry> Records sound objects.

<search_page_object_entry> Records search page objects.

<movie_object_entry> Records movie objects.

<dict_data_object_entry> Records dictionary data objects.

An example is shown in Figure A.38.

```
<parts_module>
        <object_table>
                <dynamic_text_object_entry object_id="OBT001" ... >
                        ...
                </dynamic_text_object_entry>
                <dynamic_text_object_entry object_id="OBT002" ... >
                        ...
                </dynamic_text_object_entry>
                <sound_object_entry object_id="OBS001" ...>
                        ...
                </sound_object_entry>
                <dynamic_text_object_entry object_id="OBT003" ... >
                        ...
                </dynamic_text_object_entry>
                ...
                <search_page_object_entry object_id="OBSP01" ... >
                        ...
                </search_page_object_entry>
                ...
                <movie_object_entry object_id="OBMV01" ... />
                ...
        </object_table>
</parts_module>
```

Figure A.38 – Example of the parts_module element with its descendants

A.4.5.2 Dynamic text object <dynamic_text_object_entry>

Records text objects to be used as part of the flowing content. This element has the following child element and attributes. Its syntax is given in Relax NG compact format below and explained in the following text.

```
dynamic_text_object_entry =
  element dynamic_text_object_entry {
    attlist_dynamic_text_object_entry, permission_info?
  }
attlist_dynamic_text_object_entry &=
  attribute src { text },
  attribute type { text },
```

```
attribute object_id { text }
```

[Attributes]

src: Filename of the text object instance. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

type: Stores the MIME type of the text object instance. The only possible value is "text/x-bvf-text". Shall not be omitted.

object_id: Assigns an ID number to the object. This number is used from the flow data or event data modules to refer to this object. Written in the standard object_ID format. Shall not be omitted.

[Child elements]

<permission_info> Defines the permissions concerning the object pointed to by the src attribute. Written in the standard permission format. May be omitted. The details are given in A.3.23.

Examples are shown in Figure A.39.

```
<dynamic_text_object_entry src="sect1.xml" type="text/x-bvf-text" object_id="OB03k2"/>

<dynamic_text_object_entry src="sect1.xml" type="text/x-bvf-text" object_id="OB03k1">
        <permission_info>
                <print_permission permission="authorized"/>
        </permission_info>
</dynamic_text_object_entry>
```

Figure A.39 – Examples of the dynamic_text_object_entry element with its descendants

A.4.5.3 Sound object <sound_object_entry>

The <sound_object_entry> records sound objects. Its syntax is given in Relax NG compact format below and explained in the following text.

```
sound_object_entry =
  element sound_object_entry {
    attlist_sound_object_entry, sound_object_info
  }
attlist_sound_object_entry &=
  attribute src { text },
  attribute type { text }
```

[Attributes]

type: Records the MIME type of the sound object. In this document, the following types are permitted. Shall not be omitted.

MP3: "audio/mp3"

AAC: "audio/3gpp2"

SMAF: "application/x-smaf"

src: Filename of the sound object, written in the standard Filename format. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute. Currently, only mp3 (extension: .mp3), AAC (extension: .3g2) and SMAF (extension: .mmf) are allowed.

[Child elements]

<sound_object_info> Records information concerning the sound object. The following attribute is available. Its syntax is given in Relax NG compact format below and explained in the following text.

```
sound_object_info =
  element sound_object_info { attlist_sound_object_info, empty }
attlist_sound_object_info &= attribute object_id { text }
```

[Attribute]

object_id: Assigns an ID number to the object. This number is used by the event data module to refer to this object. Written in the standard object_ID format. Shall not be omitted.

An example is shown in Figure A.40.

```
<sound_object_entry src="foobar.mp3" type="audio/mp3">
        <sound_object_info object_id="OB143s"/>
</sound_object_entry>
```

Figure A.40 – Example of the sound_object_entry element with its descendant

A.4.5.4 Search page object <search_page_object_entry>

This records search page objects. Its syntax is given in Relax NG compact format below and explained in the following text.

```
search_page_object_entry =
  element search_page_object_entry {
    attlist_search_page_object_entry, permission_info?
  }
attlist_search_page_object_entry &=
  attribute src { text },
  attribute type { text },
  attribute object_id { text }
```

[Attributes]

src: Filename of the search page object, written in the standard Filename format. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

type: Records the MIME type of the search page object. In the current specification, only "text/x-bvf-search-page" is permitted. Shall not be omitted.

object_id: Assigns an ID number to the object. This number is used by the event data module to refer to this object. Written in the standard object_ID format. Shall not be omitted.

[Child elements]

<permission_info> Defines the permissions concerning the object pointed to by the src attribute. Written in the standard Permission format. May be omitted. The details are given in A.3.23.

Examples are shown in Figure A.41.

```
<search_page_object_entry
        src="spage1.xml" type="text/x-bvf-search-page" object_id="OBSP01" />

<search_page_object_entry
        src="spage2.xml" type="text/x-bvf-search-page" object_id="OBSP02" >
        <permission_info>
                <print_permission permission="authorized"/>
        </permission_info>
</search_page_object_entry>
```

Figure A.41 – Examples of the search_page_object_entry element with its descendants

A.4.5.5 Movie object <movie_object_entry>

This records movie objects. Its syntax is given in Relax NG compact format below and explained in the following text.

```
movie_object_entry =
  element movie_object_entry { attlist_movie_object_entry, empty }
attlist_movie_object_entry &=
  attribute src { text },
  attribute type { text },
  attribute object_id { text }
```

[Attributes]

src: Filename of the movie object, written in the standard filename format. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute. Currently, only 3GPP2 (extension: .3g2) is allowed.

type: Records the MIME type of the movie object. In this document, the following type is permitted. Shall not be omitted.

3GPP2: "video/3gpp2"

object_id: Assigns an ID number to the object. This number is used by the event data module to refer to this object. Written in the standard object_ID format. Shall not be omitted.

An example is shown in Figure A.42.

```
<movie_object_entry src="movie1.3g2" type="video/3gpp2" object_id="OBmv00"/>
```

Figure A.42 – Example of the movie_object_entry element

A.4.5.6 Dictionary data object <dict_data_object_entry>

This records text dictionary data objects. Its syntax is given in Relax NG compact format below and explained in the following text.

```
dict_data_object_entry =
  element dict_data_object_entry {
    attlist_dict_data_object_entry, permission_info?
  }
attlist_dict_data_object_entry &=
  attribute src { text },
  attribute type { text },
  attribute object_id { text }
```

[Attributes]

src: Filename of the dictionary data object instance. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

type: Stores the MIME type of the dictionary data object instance. The only possible value is "text/x-bvf-dict-data". Shall not be omitted.

object_id: Assigns an ID number to the object. This number is used from the flow data or event data modules to refer to this object. Written in the standard object_ID format. Shall not be omitted.

[Child elements]

<permission_info> Defines the permissions concerning the object pointed to by the src attribute. Written in the standard permission format. May be omitted. The details are given in A.3.23.

An example is shown in Figure A.43.

```
<parts_module>
  <object_table>
    <dict_data_object_entry src="dict1.xml" type="text/x-bvf-dict-data" object_id="OBD001">
       <permission_info>…</permission_info>
    </dict_data_object_entry>
      …
  </object_table>
</parts_module>
```

Figure A.43 – Example of the parts_module element with its descendants

A.4.6 Object instances

A.4.6.1 General

The term "object instance" is used in this annex to refer to the objects that are displayed (or played, as appropriate) by the viewer. Defined are the following object instances.

Text object instance

Dictionary data object instance

Image object instance

Sound object instance

Animation object instance

Search page object instance

Movie object instance

A.4.6.2 Text object instance <text_data>

A.4.6.2.1 Text object instances

The text object instances are stored in XML files of their own, with <text_data> as the root element. The syntax of the <text_data> element is given in Relax NG compact format below.

```
text_data =
  element text_data {
    attlist_text_data, text_default_attribute, text_body
  }
attlist_text_data &= empty
```

An example is shown in Figure A.44.

```
<text_data>
        <text_default_attribute>
        ...
        </text_default_attribute>
        <text_body>
        Two households, both alike in dignity, ...
        </text_body>
</text_data>
```

Figure A.44 – Examples of the text_data element with its descendants

A.4.6.2.2 Text default attributes <text_default_attribute>

Text default attributes store parameters to use when displaying the text. Be aware that these parameters may not always be respected, if the viewer is not able to handle the values/behaviours set in this module, or if the viewer is configured to give precedence to user settings. In addition, as was explained in A.4.3.2.3, some of the parameters defined here may also be set in the whole text flow (in the <flow_default_attribute>, a child element of <flow_entry>). These are used as default if the local ones (the ones we define here) are not set. In case both are defined, the local settings take precedence.

The syntax of the <text_default_attribute> element is given in Relax NG compact format below and explained in the following text.

```
text_default_attribute =
  element text_default_attribute {
    attlist_text_default_attribute,
    text_default_font?,
    text_default_background?,
    text_default_background_music?,
    line_breaking_method?
  }
attlist_text_default_attribute &=
  attribute baseline { BaseLine }?,
  attribute valign { "middle" }?
```

[Attributes]

baseline: Defines the orientation of the baseline (and therefore of the text) for each flow. The possible values are listed below. Defaults to the <flow_default_attribute> value if omitted.

- "right": The writing direction is horizontal (left to right). However, the direction can be changed at the user's option.
- "right_only": The writing direction is horizontal (left to right), and the user cannot change the setting. However, if it is not supported by the viewer, this setting is not necessarily applied.

"down": The writing direction is vertical (top to bottom). However, the direction can be changed at the user's option.

"down_only": The writing direction is vertical (top to bottom), and the user cannot change the setting. However, if it is not supported by the viewer, this setting is not necessarily applied.

valign: Determines how the text box is to be positioned. The vertical position for horizontally written text, or the horizontal position for vertically written text is what is determined here. The value described below is eligible. When omitted, the text box should be displayed from the top of the display area.

"middle": The content of the <text_body> element is centred within the display area, as shown in Figure A.45. If the content is larger than the display area, this attribute is ignored.

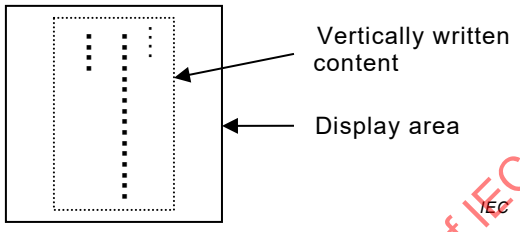


Figure A.45 – The valign attribute set to "middle"

[Child elements]

<text_default_font> Defines the default font and font colour for this text object. When omitted, defaults to the <flow_default_attribute> value. Its syntax is given in Relax NG compact format below and explained in the following text.

```
text_default_font =
  element text_default_font { attlist_text_default_font, empty }
attlist_text_default_font &=
  attribute fontname { text }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute ruby_flag { "yes" | "yes_only" | "no" | "no_only" }?
```

[Attributes]

fontname: Default font name. More than one font may be specified. In that case, each font name should be separated by a comma (U+002C). For instance:

fontname="Aaa sans serif,Bbb gothic"

The viewer should use the first listed font that is available. When omitted, defaults to the <flow_default_attribute> value.

color_space, color, opacity:

Defines the font colour to be used for the content data. Written in the standard colour data type. When omitted, defaults to the <flow_default_attribute> value.

ruby_flag: Defines whether ruby in the content data is to be viewed or not. When omitted, defaults to the <flow_default_attribute> value. The text to be displayed when ruby is turned on is the one included in the <ruby> element of text object instances. The following values may be used.

"yes": Ruby should be displayed, but can be turned off by the user.

"yes_only": Ruby shall be displayed, and cannot be turned off by the user. However, this does not apply to viewers not able to display ruby.

"no": It is recommended that ruby not be displayed, but can still be turned on according to user preferences.

"no_only": Ruby shall not be displayed, and cannot be turned off by the user. However, this does not apply if the viewer is incapable of disabling ruby display.

<text_default_ background> Defines the background colour and background image to be used for this text object. When both are defined, the image is drawn centred in the screen filled with the background colour. Finally, the content of the <text_body> element is rendered on the top. If the background image is too large to fit inside the screen, it is scaled down with the aspect ratio being preserved. The behaviour in case of omission of this element should conform to the behaviour specified in case of omission of its attributes. Its syntax is given in Relax NG compact format below and explained in the following text.

```
text_default_background =
  element text_default_background {
    attlist_text_default_background, permission_info?
  }
attlist_text_default_background &=
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute type { text }?,
  attribute src { text }?
```

[Attributes]

color_space, color, opacity:

Defines the colour to be used as background color. Written in the standard colour data type. When omitted, defaults to the <flow_default_attribute> value.

type: Stores the MIME type of the background image. May be omitted only if the src attribute is also omitted. The possible values are listed below.

"image/png"

"image/jpeg"

src: Sets the filename of the image to use as a background image, written in the standard Filename format. May be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

[Child elements]

<permission_info>:

Defines the permissions about the image referred to by the src attribute. Written in the standard Permission format. May be omitted. If the src attribute is omitted, this permission information should be ignored. The details are given in A.3.23.

<text_default_background_music>

Defines the background music to be played while displaying this text object. If this element is omitted, nothing should be played. Its syntax is given in Relax NG compact format below and explained in the following text.

```
text_default_background_music =
  element text_default_background_music {
    attlist_text_default_background_music, permission_info?
  }
attlist_text_default_background_music &=
  attribute type { text },
  attribute src { text },
  [ a:defaultValue = "no" ] attribute loop { Yes_No }?
```

[Attributes]

type: Stores the MIME type of the background music. Shall not be omitted. The possible values are listed below.

"audio/mp3"

"audio/3gpp2"

"application/x-smaf"

src: Sets the filename of the background music, written in the standard Filename format. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

loop: Specifies whether the background music should be played iteratively (i.e. repeated from the beginning every time the end is reached). Possible values are "yes" and "no". In case of omission, no is assumed.

[Child elements]

<permission_info>:

Defines the permissions about the sound file referred to by the src attribute. Written in the standard permission format. May be omitted.

<line_breaking_method> Specifies the algorithm to be used to determine how text should be split in lines. Various languages handle this in various ways, so this element allows for some flexibility. When omitted, defaults to the <flow_default_attribute> value. The syntax of <line_breaking_method> is given in Relax NG compact format below and explained in the following text.

```
line_breaking_method =
  element line_breaking_method { attlist_line_breaking_method, empty }
attlist_line_breaking_method &=
  [ a:defaultValue = "none" ] attribute method { "none" }?
```

[Attributes]

method: Chooses the line breaking method. Defaults to "none". Possible values are:

"none": no special processing, when a line is filled with characters, go to the next one.

Other values are possible for localization. See Clause A.6 for details.

Other attributes may be added by localized methods. See Clause A.6 for details.

[Child elements]

None, unless added by localized methods. See Clause A.6 for details.

An example is shown in Figure A.46.

```
<text_default_attribute>
        <text_default_font fontname="foobar-serif" color="green"/>
        <text_default_background color="black" src="001.jpg" type="image/jpeg"/>
        <text_default_background_music src="1.mp3" type="application/x-smaf"
loop="yes"/>
        <line_breaking_method ="none">
        </line_breaking_method >
</text_default_attribute>
```

Figure A.46 – Example of the text_default_attribute element with its descendants

A.4.6.2.3 Text body <text_body>

A.4.6.2.3.1 General

This part records the actual text body of the text object. In this element, Extended character string, as well as the elements described below may be used. Its syntax is given in Relax NG compact format below and explained in the following text.

```
text_body = element text_body { attlist_text_body, All_tag* }
attlist_text_body &= empty
```

A.4.6.2.3.2 Paragraph

... Creates a paragraph. Generally, there is a line break at the beginning of the paragraph (right before the element), and at its end (right after the element). However, there is no line break in the cases listed below.

If the paragraph starts at the beginning of a line even without inserting a line break.

If what follows the paragraph starts at the beginning of a line even without inserting a line break.

It should also be noted that should the paragraph end occur in text flowing around an image (see the align attribute of the <object> element), the line break does not cancel this effect. Therefore, the next line still flows around the image. In order to cancel the flowing, and start the new line after the image, <br clear="all"/> shall be explicitly used. The syntax of the element is given in Relax NG compact format below and explained in the following text.

```
p = element p { attlist_p, All_tag_p* }
attlist_p &=
  attribute top_line_indent { text }?,
  attribute top { text }?,
  attribute bottom { text }?,
  attribute align { "center" | "right" | "left" }?,
  attribute drop_cap { text }?
```

[Attributes]

top_line_indent: Sets the size (expressed in em) of the indentation of the first line of the paragraph. The spaces are inserted before the first character following the element, and the first character following a <br/> element in the paragraph. The unit is "em". The actual spacing is the sum of the value set in this attribute and the margin of the paragraph set by the "top" attribute. Moreover, this attribute may be set to a negative value, like "−2em". When omitted, defaults to 0.

top: Defines the size of the left margin of the paragraph (top margin in case the writing direction is vertical). It is expressed either in em or in a percentage of the line length (column in case the writing direction is vertical). When omitted, defaults to "0em". Negative values shall not be used. Paragraphs may be contained in another paragraph. In that case, the margins are added. Percentages are also relative to the inner area thus calculated, not the total display area.

bottom: Defines the size of the right margin of the paragraph (bottom margin in case the writing direction is vertical). It is expressed either in em or in a percentage of the total line size (column in case the writing direction is vertical). When omitted, defaults to "0em". Negative values shall not be used. Paragraphs may be contained in another paragraph. In that case, the margins are added. Percentages are also relative to the inner area thus calculated, not the total display area.

align: Determines whether the text should be in the centre, left-aligned or right-aligned within the line size defined by the

top, bottom and first line indent attributes. This attribute accepts the values listed below. When omitted, the current setting is kept unchanged.

"center": The string included in the <p> element is displayed in the centre.

"right": The string included in the <p> element is right-aligned, or bottom-aligned in the case of vertical writing.

"left": The string included in the <p> element is left-aligned, or top-aligned in the case of vertical writing.

drop_cap: Allows to turn the first letter of the paragraph into a dropped capital. The value is an integer representing the number of lines that the dropped capital should cover. An example is shown in Figure A.47. Defaults to "1" (Normal size).

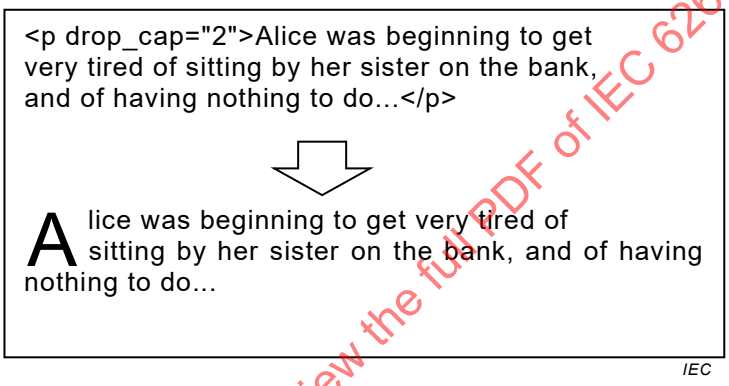


Figure A.47 – Example of dropped capital

[Child elements]

Except the <page_break/> and <head> elements, any element that may be inside <text_body>, as well as extended character strings, may be used as the content of the <p> element.

Figure A.48 shows an example of a horizontally written paragraph with top="25 %" and bottom="2em".

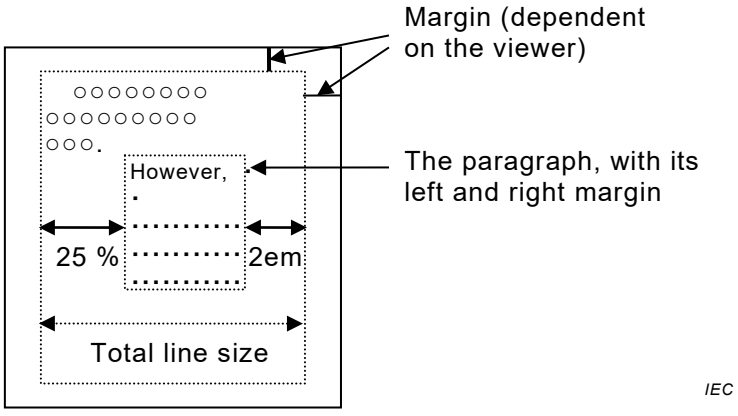


Figure A.48 – Left and right margin of a paragraph

Examples of the <p> element including the markup for above are shown in Figure A.49.

```
<p>.... </p>
<p align= "center"> ... </p>
<p top="25%" bottom="2em"> ... </p>
```

Figure A.49 – Examples of the p element

<scrolling_text> Defines a line of scrolling text. Line breaks are automatically inserted around this element (right before the <scrolling_text> element and right after the </scrolling_text> element), to guarantee that the scrolling text is alone on its line. The height of the scrolling text object depends entirely on its content. In the case of horizontally written text, scrolling is done right to left, while for vertical text it should move the text from the bottom, upwards. The text should be scrolled until it disappears from the display area (limited by the "top" and "bottom" margin of the surrounding paragraph if there is one), and then start again. The syntax of the <scrolling_text> element is given in Relax NG compact format below and explained in the following text.

```
scrolling_text =
  element scrolling_text {
    attlist_scrolling_text,
    (text
     | external_char
     | font
     | horizontal
     | ruby
     | sub
     | sup
     | object)*
  }
attlist_scrolling_text &= empty
```

[Child elements]

The string to be displayed as scrolling text is written as a child element, in the form of an Extended character string. The following elements may also be used: <external_char>, <font>, <horizontal>, <ruby>, <sub>, <sup> and <object>. However, the <object> element has the following additional limitations when used as a child element of the <scrolling_text> element:

only image files may be designated by the src attribute;

the align attribute may only be set to "top", "middle", or "bottom";

the char_id attribute shall not be used.

An example is shown in Figure A.50.

```
<scrolling_text>
        This text will be scrolling over and over
</scrolling_text>
```

Figure A.50 – Example of the scrolling_text element

A.4.6.2.3.3 Inline elements

 Inserts a line break. Its syntax is given in Relax NG compact format below and explained in the following text.

```
br = element br { attlist_br, empty }
attlist_br &= attribute clear { "left" | "right" | "all" }?
```

[Attribute]

clear: Cancels the text's flowing around an embedded object, such as an image. The <br/> element does not do so when this attribute is omitted. The following values are possible.

"left": Inserts blank lines after the <br/> element until the text can be inserted at the beginning of the line (left for horizontal writing, top for vertical). In other words, it moves the insertion point after the object, if any, around which the text flows, if it is on the left.

"right": Inserts blank lines after the <br/> element until the text can be inserted until the end of the line (right for horizontal writing, bottom for vertical). In other words, it moves the insertion point after the object, if any, around which the text flows, if it is on the right.

"all": Inserts blank lines after the <br/> element until the text can be inserted from the beginning to the end of the line. In other words, it moves the insertion point after the object, if any, around which the text flows, regardless of its being on the left or right.

Examples are shown in Figure A.51.

```
<br/>
<br clear="all"/>
```

Figure A.51 – Examples of the br element

<hr/> Draws a horizontal line (a vertical one when the text is written vertically). As this line shall always be alone in its row (column when the text is vertical), a line break (equivalent to <br/>) is always inserted after it, and before it also, unless the line starts from the top on its own. Its syntax is given in Relax NG compact format below and explained in the following text.

```
hr = element hr { attlist_hr, empty }
attlist_hr &=
  attribute size { text }?,
  attribute length { text }?,
  attribute align { "left" | "center" | "right" }?
```

[Attributes]

size:	Determines the line's thickness. Expressed in "em". Note that this value does not need to be an integer. When omitted, it defaults to "0.1em".
length:	Determines the line's length. Expressed in "em" or as a percentage of the available space (row if the text is written horizontally, column if written vertically), taking into account the enclosing <p> element's "top" and "bottom" attributes. When omitted, it defaults to "100 %".
align:	Sets the position of the line if it is drawn shorter (using the length attribute) than the available area. Possible values are "left", "center" and "right". Defaults to centre when omitted.

Examples are shown in Figure A.52.

```
<hr/>
<hr size="0.5em" length="50%" align="center"/>
```

Figure A.52 – Examples of the hr element

… Defines font properties for the text enclosed in the element. Its syntax is given in Relax NG compact format below and explained in the following text.

```
font = element font { attlist_font, Inline* }
attlist_font &=
  attribute name { text }?,
  attribute size { text }?,
  attribute base { "default" | "last" }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute bold { Yes_No }?,
  attribute underline { Yes_No }?,
  attribute italic { Yes_No }?,
  attribute oblique { Yes_No }?,
  attribute small_caps { Yes_No }?,
  attribute family {
    "monospace" | "san-serif" | "serif" | "cursive" | "fantasy"
  }?,
```

```
attribute ul_type { "rightscore" | "leftscore" | "throughscore" }?,
attribute em_type {
  "rightscore"
  | "leftscore"
  | "throughscore"
  | "kendot"
  | "bold"
  | "italic"
  | "bold_italic"
  | "reverse"
  | "shade"
}?
```

[Attributes]

name: Defines the font to use, by specifying the font name. If omitted, the font used before the <font> element will continue to apply. More than one font name may be specified. In that case, each font name should be separated by a comma (U+002C). For instance:

fontname="Aaa sans serif,Bbb gothic"

The viewer should use the first listed font that is available.

size: Sets a different font size. The size is set as a percentage of what is set by the base attribute. If omitted, the base size is not modified.

base: Defines the base of calculation for the size attribute. The values listed below are accepted. If omitted, it defaults to "last".

"last": Same size as the character before the <font> element.

"default": Same size as the default font size.

color_space, color, opacity: Changes the character's colour. Written in the standard colour data type (see A.3.16 for details). If omitted, the colour used for the characters immediately before the <font> element are used.

bold: Defines whether the string included in the <font> element should be displayed in bold face or not. The acceptable values are:

"yes": display in bold

"no": display normally

When omitted, the current state is kept.

underline: Defines whether the string included in the <font> element should be underlined or not.

"yes": display with underline

"no": display normally

Objects (e.g. image objects) included inside the <font> element should not be underlined even if this attribute is set to "yes". When omitted, the current state is kept.

italic: Decides if the content data should be displayed in italic or not. If omitted, the behaviour depends on the viewer. The acceptable values are:

"yes": display in italic

"no": display normally

When omitted, the current state is kept.

oblique: Decides if the content data should be displayed in oblique or not. If omitted, the behaviour depends on the viewer. The acceptable values are:

"yes": display in oblique

"no": display normally

When omitted, the current state is kept.

small_caps: Decides if the content data should be displayed in small capitals or not. If omitted, the behaviour depends on the viewer. The acceptable values are:

"yes": display in small capitals

"no": display normally

When omitted, the current state is kept.

family: Specifies the font family to be used. The font actually used in rendering depends on the viewer. If omitted, the behaviour is also viewer-dependent. The acceptable values are:

"monospace": display in fixed-width fonts

"sans-serif": display in fonts without serifs

"serif": display in fonts with serifs

"cursive": display in handwriting-style fonts

"fantasy": display in decorative fonts

When omitted, the current state is kept.

ul_type Specifies how strings in <u> element and its equivalent are rendered. The acceptable values are as follows, all of which are taken from ISO/IEC 9541-1:2012, 8.7.1.12.1.1. Refer to ISO/IEC 9541-1:2012 for the exact meaning of these values.

"rightscore"

"leftscore"

"throughscore"

em_type: Specifies how strings in element and its equivalent are rendered. The acceptable values are as follows, the first 4 of which are taken from ISO/IEC 9541-1:2012, 8.7.1.12.1.1. Refer to ISO/IEC 9541-1:2012 for the exact meaning of these values.

Note that when this attribute is specified, element is no longer equivalent to <font bold="yes" italic="yes">

"rightscore"

"leftscore"

"throughscore"

"kendot"

bold: The substrings should be rendered in bold font.

italic: The substrings should be rendered in italic font.

bold_italic: The substrings should be rendered in bold italic font.

"reverse": The substring should be rendered with the characters and background reversed.

"shade": The substrings should be rendered with shade.

[Child elements]

Any combination of extended character strings and all the inline elements (except <page_break/>) and <object> may be used.

Example are shown in Figure A.53.

```
<font name="Aaa sans serif">.....</font>
<font size="200%">......</font>
```

Figure A.53 – Examples of the font element

... Specifies that the substrings should be displayed in italic. Equivalent to ...

```
i = element i { attlist_i, Inline* }
attlist_i &= empty
```

<u>...</u> Specifies that the substrings should be underlined. Equivalent to .... It has the following attribute.

```
u = element u { attlist_u, Inline* }
attlist_u &= attribute ul_type { "rightscore" | "leftscore" | "throughscore" }?
```

ul_type Specifies how strings in <u> element and its equivalent should be rendered. The acceptable values are as follows, all of which are taken from ISO/IEC 9541-1:2012, 8.7.1.12.1.1. Refer to ISO/IEC 9541-1:2012 for the exact meaning of these values.

"rightscore"

"leftscore"

"throughscore"

... Specifies that the substrings should be displayed in bold italic. Equivalent to ... in default settings. Its syntax is given in Relax NG compact format below and explained in the following text.

```
em = element em { attlist_em, Inline* }
attlist_em &=
  attribute em_type {
    "rightscore"
```

```
  | "leftscore"
  | "throughscore"
  | "kendot"
  | "bold"
  | "italic"
  | "bold_italic"
  | "reverse"
  | "shade"
}?
```

If em_type attribute in <flow_default_font> and <text_default_font> is specified, this is not the case and how the strings in this element should be rendered depends on the em_type attribute.

em_type: Specifies how strings in element and its equivalent are rendered. May be omitted. When specified, this overrides what was specified by the em_type attribute of <flow_default_font>. The acceptable values are as follows, the first 4 of which are taken from ISO/IEC 9541-1:2012, 8.7.1.12.1.1. Refer to ISO/IEC 9541-1:2012 for the exact meaning of these values.

"rightscore"

"leftscore"

"throughscore"

"kendot"

"bold": The substrings should be rendered in bold font.

"italic": The substrings should be rendered in italic font.

"bold_italic": The substrings should be rendered in bold italic font.

"reverse": The substring should be rendered with the characters and background reversed.

"shade": The substrings should be rendered with shade.

<oblq>…</oblq> Specifies the substring should be displayed in oblique. Equivalent to …. Its syntax is given in Relax NG compact format below.

```
oblq = element oblq { attlist_oblq, Inline* }
attlist_oblq &= empty
```

<sc>…</sc> Specifies the substring should be displayed in small capitals. Equivalent to …. Its syntax is given in Relax NG compact format below.

```
sc = element sc { attlist_sc, Inline* }
attlist_sc &= empty
```

<mlg>…</mlg> Represents a multiline note. Although this annex does not specify how multiline notes should be rendered, an example of how they are to be rendered is shown below for a better understanding of the concept.

An example is shown in Figure A.54

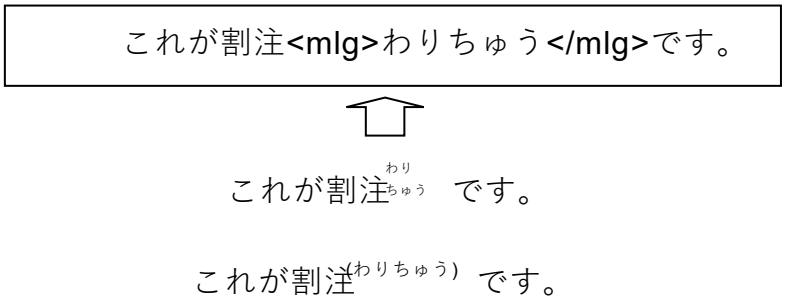


Figure A.54 – Example of the mlg element and examples of how it is rendered

The syntax of the <mlg> element is given in Relax NG compact format below.

```
mlg = element mlg { attlist_mlg, Inline* }
attlist_mlg &= empty
```

<horizontal>…</horizontal> Defines substrings that should be displayed horizontally even when the general text direction is vertical. This is often used for dates and other numbers in Japanese texts, see Figure A.55 for examples. When the text is written horizontally, this element has no effect. Its syntax is given in Relax NG compact format below and explained in the following text.

```
horizontal = element horizontal { attlist_horizontal, TextWithGaiji }
attlist_horizontal &= empty
```

[Child elements]

External extended character strings may be used here.

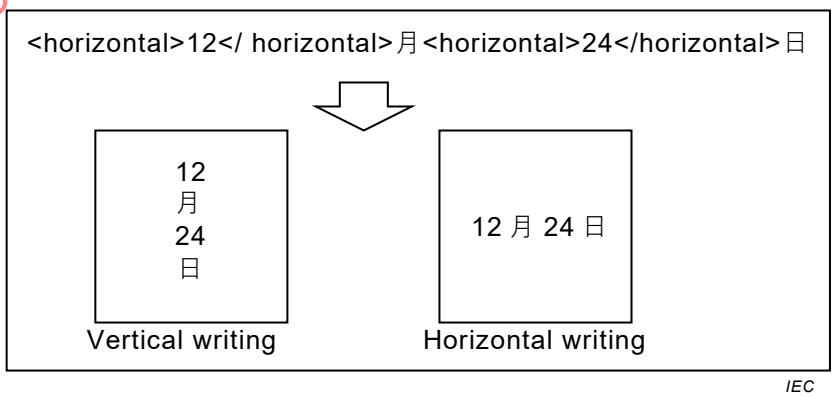


Figure A.55 – Examples of horizontal writing in vertical text

<ruby>…</ruby> Used to display ruby text. Ruby is an often used feature in Asian, and particularly Japanese typography. It consists in a short run of text displayed above the base text, mostly used as a pronunciation guide for ideographic characters. As far as the available fonts permit it, ruby text is usually displayed in smaller characters than the base text it annotates. See Figure A.56 for an example. How line spacing is handled to open enough space to display ruby depends on the viewer. It should also be noted that as ruby is fundamentally a reading aid, and some people may not want or need it, some viewers may have an option to disable ruby display altogether. The syntax of the <ruby> element is given in Relax NG compact format below and explained in the following text.

```
ruby = element ruby { attlist_ruby, rbase, rtop }
attlist_ruby &= empty
```

[Child elements]

<rb> The base string which should be annotated by a ruby. Shall not be omitted. Consists of an External extended character string. Its syntax is given in Relax NG compact format below.

```
rb = element rb { attlist_rb, TextWithGaiji }
attlist_rb &= empty
```

<rt> The ruby string that will be displayed above the base (or on the right side, if displaying vertical text). Shall not be omitted. Consists of an External extended character string. Its syntax is given in Relax NG compact format below.

```
rt = element rt { attlist_rt, TextWithGaiji }
attlist_rt &= empty
```

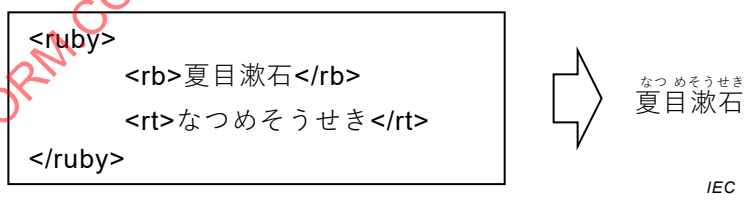


Figure A.56 – Example of the ruby element with descendants and how they are to be typically rendered

_… Represents the substring as subscripts. Its syntax is given in Relax NG compact format below.

```
sub = element sub { attlist_sub, TextWithGaiji }
attlist_sub &= empty
```

[…] Represents the substring as superscripts. Its syntax is given in Relax NG compact format below.

```
sup = element sup { attlist_sup, TextWithGaiji }
attlist_sup &= empty
```

<external_char> Inserts an external character. The syntax is described in detail in A.3.11. Within the text object instance, the alt_img and alt_vimg attributes may be used.

An example is shown in Figure A.57.

```
<external_char alt_set="oooextchars" alt_code="0x1234"
    alt_img="ou.img" alt_vimg="ou_v.img"
    img_type="image/jpeg" alt="鴎"/>
```

Figure A.57 – Example of the external_char_element

<mask> Adds a masking capability to the string embedded in this element. When this string or one designated by the trigger attribute is clicked, the child element string is covered and uncovered alternately. Its syntax is given in Relax NG compact format below and explained in the following text.

```
attlist_mask &=
  [ a:defaultValue = "on" ] attribute initial_flag { "on" | "off" }?,
  attribute trigger { text }?,
  attribute char_id { text }?,
  [ a:defaultValue = "default" ]
  attribute mask_type { "default" | "color" }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  [ a:defaultValue = "black" ] attribute color { text }?,
  [ a:defaultValue = "scope" ] attribute hold_flag { HoldFlag }?
```

[Attributes]

initial_flag: Determines the initial state of the mask. The possible values are listed below, with "on" being the default in case of omission.

"on": The content is hidden by the mask.

"off": The content is not hidden.

trigger: Designates the string(s) that trigger the mask's status change. When a string indicated by this attribute is clicked, the mask should toggle between on and off. Written as one or more Char_IDs with comma signs (U+002C) separating them in the latter case, as seen in the following.

Examples are shown in Figure A.58.

```
"CR0001,CR0002,CR0003"
```

Figure A.58 – Examples of Char_ID

The designated character string shall be in the same text object instance. Note that the char id used here shall not be specified at the same time as a trigger in the event info module. If it is specified, no action is triggered on the mask even if the designated string is clicked. When omitted, the string enclosed in the mask itself serves as a trigger.

char_id: Sets a string ID to this <mask> object. Written in the standard Char_ID format. May be omitted.

mask_type: Selects the type/style of mask to be used. The values listed below may be used. Defaults to "default" when omitted.

"default": The apperance of the mask depends on the viewer.

"color": The mask is a filled, colored box. The colour is either defined by the corresponding attributes, or their default value in case of omission.

color_space, color, opacity: Defines the colour of the mask, using the standard Color data type, as described in A.3.16. Shall be defined only if the mask_type attribute is set to "color". If omitted, defaults to "black".

hold_flag: Defines the state recording policy, allowing to remember whether the mask has been turned on or off. May be set to the values listed below. Defaults to "scope" when omitted.

"scope": Remembers the state as long as the current text object instance is displayed.

"on_power": Remembers the state as long as the document is open.

"save": The state should be saved when closing the document and restored next time it is opened.

[Child elements]

The content of this element may be any combination of external extended character strings and the <br>, <font>, <horizontal>, <ruby> and <object> elements. However, the following restrictions apply to the <object> element:

The src attribute shall point to an image file.

The char_id attribute shall not be used.

Examples are shown in Figure A.59.


```
The answer is <mask>42 </mask>.
The answer is <mask trigger ="CR0002">42</mask>.
```

Figure A.59 – Examples of the mask element

<char_id> </char_id> Attaches a string ID number to its content. Its syntax is given in Relax NG compact format below and explained in the following text.

```
char_id = element char_id { attlist_char_id, Inline* }
attlist_char_id &= attribute char_id { text }
```

[Attribute]

char_id: Sets the ID number of the element's content, written in the Char_ID standard format. Shall not be omitted.

[Child elements]

The content of this element may be any combination of Extended character strings, inline elements except <page_break>, and the <object> element.

<head> </head> Defines a list of keywords to be used for searching capabilities. The following child elements are available. It is possible to write the string right after the </head> element. The interpretation for such a description is left to the viewer and the conversion (if any). Its syntax is given in Relax NG compact format below and explained in the following text.

```
head = element head { attlist_head, headword+, key* }
attlist_head &= empty
```

[Child elements]

<headword> Records a single keyword. There shall be at least one <headword> per <head>. The <headword> element can also be used for regisitering the word for inclusion in a search table by specifying table_id attribute. Its syntax is given in Relax NG compact format below and explained in the following text.

```
headword = element headword { attlist_headword, TextWithGaiji }
attlist_headword &=
  attribute type { text }?,
  attribute table_id { text },
  [ a:defaultValue = "IPA" ] attribute phonetic_notation { text }?
```

[Attributes]

type: specifies the type of the word in a standard character string. The following values are pre-defined while other user-defined values are allowed as well. However, it is strongly recommended that "pronunciation" be used when applicable (i.e. the string represents phonetic symbols).

"pronunciation": The string represents pronunciation symbols.

other values: Other user-defined values are possible.

table_id: Points to the search table to which this keyword is to be added, by using the table's ID number, which is defined in <seach_table_def> (see A.4.3.2.6). Shall not be omitted if the content has a search page object (see A.4.6.7). Not to be

confused with Reference ID (see A.4.6.3.4).

phonetic_notation: Denotes the phonetic alphabet used. Defaults to "IPA". To be ignored when the type is not "pronunciation". The following values are pre-defined while other user-defined values are allowed as well. However, it is strongly recommended that "IPA" be used when reasonably applicable.

"IPA": Denotes "International Phonetic Alphabet",

[Child elements]

Inside the element is stored the text that should be displayed in the search results when the keyword matches. It is simply an External extended character string.

<key> </key> Defines the actual keyword. Only the characters allowed in the corresponding search table may be used if <search_table> exists. See <enable_key_type> for details in A.4.3.2.6. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
key = element key { attlist_key, text }
attlist_key &= attribute type { text }?
```

[Attribute]

type: Specifies the type of the keyword in Standard character string. May be omitted.

An example of the head element with its descendants and the subsequent meaning element is shown in Figure A.60.

```
<head>
        <headword table_id="ST0001">color</headword>
                <key>color</key>
                <key>colour</key>
        <headword table_id="ST0002">color,colour</headword>
                <key>カラー</key>
</head>
<meaning> Visual effect caused by the wavelength of light…
```

Figure A.60 – Example of the head element with its descendants and the meaning element

<meaning> </meaning> Gives definitions. Its syntax is given in Relax NG compact format below and explained in the following text.

```
meaning = element meaning { attlist_meaning, All_tag_p* }
attlist_meaning &=
```

```
  attribute type { text }?,
  attribute subid { text }?,
  attribute level { text }?,
  attribute no { text }?
```

[Attributes]

type: Specifies the category of the data. There are no predefined categories.

subid: Specifies a Reference ID to refer to a position in the definition.

level: Specifies the hierarchical level of the meaning in numerals [0-9]. Recorded in the order of importance (i.e. the smaller, the nearer to the top of the hierarchy) though it is not required that the values be consecutively used.

no: Specifies the number of the meaning in the associated level in numerics [0-9].

<example> </example> Gives examples. Its syntax is given in Relax NG compact format below and explained in the following text.

```
example = element example { attlist_example, All_tag_p* }
attlist_example &= empty
```

When an example is given in bilingual format, it can either be represented using elements with the lang attribute set to corresponding languages, or as a plain text with a delimiter implicitly predefined as appropriate.

An example is shown in Figure A.61.

```
<example>
<span xml:lang="en">I'll take eel.</span>
<span xml:lang="ja">僕はうなぎだ。</span></example>
<example>I'll take eel|僕はうなぎだ。</example>
```

Figure A.61 – Examples of the example element with its descendants

<subhead> </subhead> Defines sub-headwords and related data. Intended to be used for introducing derived and compound words, etc. Its syntax is given in Relax NG compact format below and explained in the following text.

```
subhead =
  element subhead {
    attlist_subhead, subheadword+, meaning*, example*, key*
  }
attlist_subhead &=
  attribute type { text }?,
  attribute subid { text }?
```

[Attributes]

subid: Defines a Reference ID for this entry. May be omitted.

type: Defines type for the subhead word (e.g. derived word, compound word) in a Standard character string. May be omitted.

[Child elements]

<subheadword> </subheadword> Defines sub-headwords. Its syntax is given in Relax NG compact format below and explained in the following text.

```
subheadword = element subheadword { attlist_subheadword, TextWithGaiji }
attlist_subheadword &=
  attribute type { text }?,
  attribute subid { text }?
```

[Attributes]

subid: Defines a Reference ID for the subhead word. May be omitted.

type: Defines type for the subhead word (e.g. derived word, compound word) in a standard character string. The following values are pre-defined while other user-defined values are allowed as well. However, it is strongly recommended that "pronunciation" be used when applicable. May be omitted.

"pronunciation": The string represents pronunciation symbols.

[Child elements]

Inside the element is stored the text that should be displayed in the search results when the keyword matches. It is simply an External extended character string.

<key> </key> See the explanation of the <key> element above for details.

<meaning> See the explanation of the <meaning> element above for details.

<example> Gives definitions. See the explanation of the <example> element above for details.

<ref> </ref> Provides reference to the other items. Its syntax is given in Relax NG compact format below and explained in the following text.

```
ref = element ref { attlist_ref, All_tag_p* }
attlist_ref &=
  attribute type { text }?,
  attribute refid { text }
```

[Attributes]

refid: Specifies the id or subid attribute value of the referenced element. Cannot be omitted.

type: Specifes the type of reference.

[Child elements]

The same as <p>.

<split> </split> Specifies the delimiter between lexicographical blocks (e.g. this element may be used between the last word starting with "A" and the first word starting with "B"). This element is intended to record the starting points of such blocks as well as what strings should be rendered for that (e.g. "-B-" for a block consisting of words starting with "B"). Its syntax is given in Relax NG compact format below and explained in the following text.

```
split = element split { attlist_split, All_tag_p* }
attlist_split &= attribute level { text }?
```

[Attribute]

level: Defines hierarchical levels in numerals [0-9]. Recorded in the order of hierarchy (i.e. the smaller, the nearer to the top of the hierarchy) though it is not required that the values be consecutively used.)

[Child elements]

The same as <p>.

Examples are shown in Figure A.62.

```
<dict_body>
<split>- A -</split>
<dic-item> … </dic-item>
<dic-item> … </dic-item>
        :
<dic-item>…</dic-item>
<split>- B -</split>
<dic-item>…</dic-item>
        :
<dict_body>
```

Figure A.62 – Examples of the split elements

<column> </column> Introduces articles. Its syntax is given in Relax NG compact format below and explained in the following text.

```
column = element column { attlist_column, All_tag_p* }
attlist_column &=
  attribute type { text }?,
  attribute subid { text }
```

[Attributes]

subid: Defines a Reference ID for the column.

type: Defines the type of the column in a Standard character string.

[Child elements]

The same as <p>.

<page_break/> Inserts a page break. This element has attributes listed below. It shall not be used as a child element of any of the elements that may be used inside <text_body>. Note that when <page_break/> is the last element of <text_body>, a blank page is inserted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
page_break = element page_break { attlist_page_break, empty }
attlist_page_break &= attribute turning_page_control { Turn_Page_Val }?
```

[Attribute]

turning_page_control: Defines how ordinary scrolling interacts with the page breaks, by allowing or forbidding crossing the page break forward or backward. The possibilities are listed below.

"on": The page break shall not be crossed by scrolling.

"off": The page break can be crossed by scrolling.

"forward": The page break can be crossed by scrolling only when moving forward.

"back": The page break can be crossed by scrolling only when moving backward.

When omitted, the behaviour should conform to the global setting defined for the text object instance (the turning_page_control attribute of <flow_data>, see A.4.3.2.4).

Examples are shown in Figure A.63.

```
<page_break/>
<page_break turning_page_control="off"/>
```

Figure A.63 – Examples of the page_break element

A.4.6.2.3.4 Object

<object>...</object> Inserts an external object in the text body. Its syntax is given in Relax NG compact format below and explained in the following text.

```
object = element object { attlist_object, permission_info? }
```

```
attlist_object &=
  attribute type { text },
  attribute src { text },
  attribute char_id { text }?,
  attribute align { "top" | "middle" | "bottom" | "left" | "right" }?,
  attribute start { "auto" | "event" }?,
  attribute loop { "1" }?
```

[Attributes]

type: Specifies the type of the inserted object, by giving the MIME type of the file. Shall not be omitted. In this edition of IEC 62605, the following types are permitted:

- PNG image: "image/png"
- JPEG image: "image/jpeg"
- MP3 audio data: "audio/mp3"
- Animation object: "application/x-bvf-flip-animation" (see A.4.6.6)

src: Sets the filename of the object to be inserted, written in the standard Filename format. Shall not be omitted. It shall be checked against the type attribute before the file referred to by this src attribute is read.

char_id: Assigns a string ID number to this object, written as a Char_ID. May be omitted.

align: Defines how text should flow around the inserted object. The possible values are listed below. When omitted, it defaults to bottom for horizontally written text, and to middle for vertically written text.

"top": Inline display. When text is written horizontally, the topmost part of the row should be aligned with the upper edge of the object; when written vertically, the rightmost part of the column should be aligned with the right edge of the object. Figure A.64 illustrates the behaviour in the former case.

Figure A.64 – The align attribute set to "top"

"middle": Inline display. When text is written horizontally, the object should be vertically centred on the row's base line. When writing vertically, the object should be horizontally centred on the column's base line. Figure A.65 illustrates the behaviour in the former case.

Figure A.65 – The align attribute set to "middle"

"bottom": Inline display. When text is written horizontally, the base line of the row should be aligned with the lower edge of the object; when written vertically, the base line of the column should be aligned with the left edge of the object. Figure A.66 illustrates the behaviour in the former case.

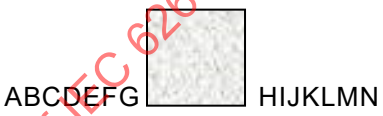


Figure A.66 – The align attribute set to "bottom"

"left": Flow around. When displaying horizontal text, the object is displayed next to the left margin, and the text is written in the space between the object and the right margin. When writing vertically, the object is displayed at the top, and text below it. Figure A.67 illustrates the behaviour in the former case.

If the object is larger than the space for one row/column, there may be more than one rows/column of text beside it. The align attribute of the surrounding paragraph (the <p> element) does not influence how the object is positioned. Even if set to something different from "left", the object is displayed along the left margin. The paragraph settings, however, still apply to the text flowing around the object. When the <object> element appears on the middle of a text row, it is displayed at the beginning of the next line.

The following example should be rendered the same way with the <object> element at any position between after the letter "A" and before the letter "M".

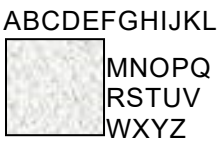


Figure A.67 – The align attribute set to "left"

"right": Flow around. When displaying horizontal text, the object is displayed next to the right margin, and the text is written in the space between the object and the left margin. When writing vertically, the object is displayed at the bottom, and the text is above it. Figure A.68 illustrates the behaviour in the former case.

If the object is larger than the space for one row/column, there may be more than one row/column of text beside it.

The align attribute of the surrounding paragraph (the <p> element) does not influence how the object is positioned. Even if set to something different from "right", the object is displayed along the right margin. The paragraph settings, however, still apply to the text flowing around the object. When the <object> element appears on the middle of a text row, it is displayed on the right of the next line.

The following example should be rendered the same way with the <object> element at any position between after the letter "A" and before the letter "M".

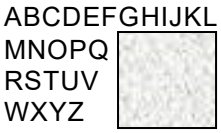


Figure A.68 – The align attribute set to "right"

start: When the src attribute points to an animation object, this allows to define when the playback is started. Can only be used with animation objects. The possible values are listed below. When omitted, defaults to "auto". Moreover, when set to auto, the animation shall not be used by the <action_play> (see A.4.4.4.2) element.

"auto": Playback starts automatically, as soon as the object is displayed, running the animation from the start. If the animation is hidden and displayed again (because of scrolling), it should start again from the beginning, regardless of where it stopped.

"event": Playback is handled according to event data (see A.4.4.2).

loop: Sets the number of times an animation should be played. In this document, the only accepted value is "1". When omitted, the animation object is played infinitely looping, until it goes out of the display area. However, if the start attribute is not set to "auto", the value of this loop attribute is ignored, and the behaviour should be as if it were set to "1".

[Child elements]

<permission_info> Defines the permissions about the object referred to by the src attribute. Written in the standard permission format. May be omitted. The details are given in A.3.23.

Examples are shown in Figure A.69.

```
<object type="image/png" src="image1.png"/>
<object type="image/jpeg" src="image2.jpg">
        <permission_info>
                <print_permission permission="authorized"/>
        </permission_info>
</object>
<object type="application/x-bvf-flip-animation"
                src="anime.xml" loop="1"/>
```

Figure A.69 – Examples of the object element

A.4.6.2.3.5 Logical block elements

... Represents a logical block. May be nested. Its syntax is given in Relax NG compact format below and explained in the following text.

```
\div = element div { attlist_div, All_tag_p* }
attlist_div &=
  attribute level { text }?,
  attribute title { text }?
```

[Attributes]

level: Records the depth of the level in numerals [0-9]. May be omitted. Recorded in the order of importance (i.e. the smaller, the nearer to the top of the hierarchy) though it is not required that the values be consecutively used.

title: Records the title for the block. May be omitted. Intended to record the information.

[Child elements]

The same as <p>.

... An element to be used to distinguish substrings that are not to be reflected in the final rendering, such as editor's memorandum. It is to be ignored by the viewer. This is also intended to be used as delimiters in <example> elements. Its syntax is given in Relax NG compact format below and explained in the following text.

```
span = element span { attlist_span, All_tag_p* }
attlist_span &= attribute type { text }?
```

[Attributes]

type: Specifies the type of the substring. In standard character string. May be omitted.

[Child elements]

The same as <p>.

A.4.6.2.3.6 Table elements

<table>…</table> Introduces an HTML-like table. This is meant to arrange data into rows and columns on rendering, and it is not to be confused with search tables (see A.4.3.2.6). Its syntax is given in Relax NG compact format below and explained in the following text.

```
table = element table { attlist_table, tr+ }
attlist_table &= empty
```

[Child elements]

<tr>…</tr> Defines rows in the table. <table> needs to have at least one instance of the <tr> element as a child element. Its syntax is given in Relax NG compact format below.

```
tr = element tr { attlist_tr, td+, th* }
attlist_tr &= empty
```

[Child elements]

<td>…</td> Defines cells in the row. <tr> needs to have at least one instance of the <td> element as a child element. Its syntax is given in Relax NG compact format below.

```
td = element td { attlist_td, Inline* }
attlist_td &= empty
```

<th>…</th> Defines a header for the table. May be omitted. Its syntax is given in Relax NG compact format below.

```
th = element th { attlist_th, Inline* }
attlist_th &= empty
```

A.4.6.2.3.7 Display according to the environment

<select>…</select> Represents alternative strings to be displayed according to the environment. Its syntax is given in Relax NG compact format below and explained in the following text.

```
select = element select { attlist_select, select_item+ }
attlist_select &=
  attribute default { text }?,
```

attribute type { text }?

[Child elements]

<select_item> Defines strings for each environment. Its syntax is given in Relax NG compact format below and explained in the following text.

select_item = element select_item { attlist_select_item, All_tag_p* }

attlist_select_item &= empty

[Attributes]

default: Specifies that the instance of the element gives the default environment when this attribute is set to "yes." Otherwise it is ignored.

type: Specifies the type of the environment (e.g. the name of the hardware for which the content is intended.) in a standard character string. Shall not be omitted.

A.4.6.2.3.8 TTS (text-to-speech) functionality

<tts>…</tts> Specifies the substring to be fed into text to speech representation. Its syntax is given in Relax NG compact format below and explained in the following text.

tts = element tts { attlist_tts, All_tag_p* }

attlist_tts &= attribute speaker { "male" | "female" | "child" }?

[Attribute]

speaker: Specifies the type of the speaker. Accepted values are:

"male": grown-up male

"female" grown-up female

"child" child

A.4.6.2.3.9 Event

<a> Specifies an anchor to the link in a similar manner to the corresponding <a> element of HTML. Its syntax is given in Relax NG compact format below and explained in the following text.

a = element a { attlist_a, Inline* }

attlist_a &= attribute href { text }

[Attribute]

href: Specifies the location of a Web resource.

A.4.6.3 Dictionary data object instance

A.4.6.3.1 General

The dictionary data object instances are stored in XML files of their own, with the <dict_data> element as the root element. Its syntax is given in Relax NG compact format below and explained in the following text.

```
dict_data =
  element dict_data {
    attlist_dict_data, dict_default_attribute, dict_body
  }
attlist_dict_data &= empty
```

[Child elements]

<dict_default_attribute> The details are given in A.4.6.3.2.

<dict_body> The details are given in A.4.6.3.3.

[Attribute]

None.

As can be seen, the <dict_data> element has two main child elements, one for storing the dictionary data's default attributes (the <dict_default_attribute>), and one to store the data itself (<dict_body>).

A.4.6.3.2 Dictionary default attributes <dict_default_attribute>

Dictionary default attributes store parameters to use when displaying the dictionary data. Be aware that these parameters may not always be respected, if the viewer is not able to handle the values/behaviors set in this module, or if the viewer is configured to give precedence to user settings. In addition, as was explained in A.4.3.2.3, some of the parameters defined here may also be set in the whole text flow (in the <flow_default_attribute> child element of <flow_entry> see A.4.3.2.4). These are used as default if not set locally as shown below. In case both are defined, the local settings take precedence.

The syntax of the <dict_default_attribute> element is given in the RelaxNG compact format below and explained in the following text.

```
dict_default_attribute =
  element dict_default_attribute {
    attlist_dict_default_attribute,
    dict_default_font?,
    dict_default_background?,
    dict_default_background_music?,
    line_breaking_method?
  }
```

```
attlist_dict_default_attribute &=
  attribute baseline { BaseLine }?,
  attribute valign { "middle" }?
```

[Attributes]

baseline: Defines the orientation of the baseline (and therefore of the text) for each flow. The possible values are listed below. May be omitted. If neither this attribute nor the corresponding option in the <flow_default_attribute> (see A.4.3.2.3 for details) is set, the default value depends on the viewer.

- "right": The writing direction is horizontal (left to right). However, the direction can be changed at the user's option.
- "right_only": The writing direction is horizontal (left to right), and the user cannot change the setting. However, if it is not supported by the viewer, this setting is not necessarily applied.
- "down": The writing direction is vertical (top to bottom). However, the direction can be changed at the user's option.
- "down_only": The writing direction is vertical (top to bottom), and the user cannot change the setting. However, if it is not supported by the viewer, this setting is not necessarily applied.

valign: Determines how the text box is to be positioned. The vertical position for horizontally written text, or the horizontal position for vertically written text is what is determined here. The value described below is eligible. When omitted, the text box should be displayed from the top of the display area.

- "middle": The content of the <text_body> element is centred within the display area. See Figure A.45 in A.4.6.2.2. If the content is larger than the display area, this attribute is ignored.

[Child elements]

<dict_default_font> Defines the default font and font colour for this dictionary data object. When omitted, defaults to the <flow_default_attribute> value. It has the following attribute. Its syntax is given in Relax NG compact format below and explained in the following text.

```
dict_default_font =
  element dict_default_font { attlist_dict_default_font, empty }
  attlist_dict_default_font &=
  attribute fontname { text }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute ruby_flag { "yes" | "yes_only" | "no" | "no_only" }?
```

[Attributes]

fontname: Default font name. More than one font may be specified. In that case, each font name should be separated by a comma (U+002C). For instance:

fontname="Aaa sans serif,Bbb gothic"

The viewer should use the first listed font that is available. When omitted, defaults to the <flow_default_attribute> value.

color_space, color, opacity:

Defines the font colour to be used for the content data. Written in the standard colour data type. When omitted, defaults to the <flow_default_attribute> value.

ruby_flag: Defines whether ruby in the content data is to be viewed or not. When omitted, defaults to the <flow_default_attribute> value. The text to be displayed when ruby is turned on is the one included in the <ruby> element of dictionary data object instances. The following values may be used.

"yes": Ruby should be displayed, but can be turned off by the user.

"yes_only": Ruby shall be displayed, and cannot be turned off by the user. However, this does not apply to viewers not able to display ruby.

"no": It is recommended that ruby not be displayed, but can still be turned on according to user preferences.

"no_only": Ruby shall not be displayed, and cannot be turned off by the user. However, this does not apply if the viewer is incapable of disabling ruby display.

<dict_default_ background> Defines the background colour and background image to be used for this dictionary data object. When both are defined, the image is drawn centred in the screen filled with the background colour. Finally, the content of the <dict_body> element is rendered on the top. If the background image is too large to fit inside the screen, it is scaled down with the aspect ratio being preserved. The behaviour in case of omission of this element should conform to the behaviour specified in case of omission of its attributes. Its syntax is given in Relax NG compact format below and explained in the following text.

```
dict_default_background =
  element dict_default_background {
    attlist_dict_default_background, permission_info?
  }
attlist_dict_default_background &=
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute type { text }?,
  attribute src { text }?
```

[Attributes]

color_space, color, opacity:

Defines the colour to be used as background color. Written in the standard colour data type. When omitted, defaults to the <flow_default_attribute> value.

type: Stores the MIME type of the background image. May be omitted only if the src attribute is also omitted. The possible values are listed below.

"image/png"

"image/jpeg"

src: Sets the filename of the image to use as a background image, written in the standard Filename format. May be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

[Child elements]

<permission_info>:

Defines the permissions about the image referred to by the src attribute. Written in the standard Permission format. May be omitted. If the src attribute is omitted, this permission information should be ignored. The details are given in A.3.23.

<dict_default_background_music>

Defines the background music to be played while displaying this dictionary data object. If this element is omitted, nothing should be played. Its syntax is given in Relax NG compact format below and explained in the following text.

```
dict_default_background_music =
  element dict_default_background_music {
    attlist_dict_default_background_music, permission_info?
  }
attlist_dict_default_background_music &=
  attribute type { text },
  attribute src { text },
  [ a:defaultValue = "no" ] attribute loop { Yes_No }?
```

[Attributes]

type: Stores the MIME type of the background music. Shall not be omitted. The possible values are listed below.

"audio/mp3"

src: Sets the filename of the background music, written in the standard filename format. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

loop: Specifies whether the background music should be played iteratively (i.e. repeated from the beginning every time the end is reached). Possible values are "yes" and "no". In case of omission, no is assumed.

[Child elements]

<permission_info>:

Defines the permissions about the sound file referred to by the src attribute. Written in the standard permission format. May be omitted. The details are given in A.3.23.

<line_breaking_method> The details are given in A.4.6.2.2.

A.4.6.3.3 Main part of the dictionary data <dict_body>

The <dict_body> element stores the main part of the dictionary data. Its syntax is given in Relax NG compact format below and explained in the following text.

```
dict_body = element dict_body { attlist_dict_body, split*, dic-item+ }
attlist_dict_body &= empty
```

[Child elements]

<dic-item> The details are given in A.4.6.3.4.

<split> The details are given in A.4.6.2.3.

[Attributes]

None.

A.4.6.3.4 Entry item <dic-item>

The <dic-item> element records information for each individual entry. The same number of the elements exist as words defined. At least one instance of this element shall appear in a content. Its syntax is given in Relax NG compact format below and explained in the following text.

```
dic-item =
  element dic-item {
    attlist_dic-item,
    (All_tag_di
     | gender
     | psp
     | glabel
     | pronunciation
     | inflec
     | slabel
     | guideword
     | spellout
     | variant
     | etymology)*
  }
attlist_dic-item &=
```

```
attribute type { text }?,
attribute id { text },
attribute rank { text }?,
attribute level { text }?,
[ a:defaultValue = "no" ] attribute page_break { Yes_No }?,
attribute turning_page_control { Turn_Page_Val }?,
attribute revision { text }?,
attribute delete { Yes_No }?
```

[Attributes]

type: Specifies the type of the entry in standard character string. May be omitted.

id: Specifies an identifier (Reference ID) for this entry. Reference ID is used for:

a) accessing the item from the other parts of the content;

b) identifying the item in editing and updating the content.

Shall not be omitted. It is an alphanumeric character string and may be an empty string ("").

rank: Represents the importance level for this entry (e.g. for learners) in numerals [0-9].

level: Records the hierarchy of entries (e.g. chapter-section hierarchy), which is otherwise lost. To be specified in numerals [0-9]. Recorded in the order of importance (i.e. the smaller, the nearer to the top of the hierarchy) though it is not required that the values be consecutively used.)

page_break: Inserts a page break. When omitted, it defaults to "no."

"yes": insert a page break

"no": do not insert a page break

turning_page_control: Defines how ordinary scrolling interacts with the page breaks, by allowing or forbidding crossing the page break forward or backward. The possibilities are listed below.

"on": The page break shall not be crossed by scrolling.

"off": The page break can be crossed by scrolling.

"forward": The page break can be crossed by scrolling only when moving forward.

"back": The page break can be crossed by scrolling only when moving backward.

When omitted, the behaviour should conform to the global setting defined for the dictionary data object instance (the turning_page_control attribute of <flow_data>, see A.4.3.2.4).

revision: Specifies the revision number corresponding to the instance of this element in a numeral not smaller than 1. May be omitted. When omitted, it is interpreted as no revision, i.e. the first version of the content.

delete: Specifies if this element should be removed due to revision. The following values are permitted. Once a certain instance of this element was deleted setting its delete attribute to "yes", a new instance of <dic-item> element should be added when it is restored in later revisions. When omitted, it defaults to "no".

"yes": deleted

"no": not deleted

[Child elements]

As shown in Relax NG compact format above.

Note the following:

- <head> shall appear exactly once at the top of inside <dic-item>;
- in <head>,<headword> shall appear at least once;
- if the <headword> element appears more than once, bookmarks and wordbook functions are applicable to the first one only;
- <external_char> shall not appear inside <headword>.

[Child elements]

Only those that are specific to the <dic-item> elements are shown below.

<gender> Gramatical gender info. <psp> can include such information without using this element, depending the policy of the publisher. Its syntax is given in Relax NG compact format below.

```
gender = element gender { attlist_gender, Inline* }
attlist_gender &= empty
```

<glabel> Denotes grammatical label (e.g. ":AmE", "Abbrev."). Its syntax is given in Relax NG compact format below.

```
glabel = element glabel { attlist_glabel, Inline* }
attlist_glabel &= empty
```

<pronunciation> Records a phonetic alphabet string. Its syntax is given in Relax NG compact format below and explained in the following text. This element is intended for recording phonetic alphabet strings outside <headword> and <subheadword> while such usage is not prohibited. The following attributes are possible. Its syntax is given in Relax NG compact format below and explained in the following text.

```
pronunciation = element pronunciation { attlist_pronunciation, Inline* }
attlist_pronunciation &=
```

[a:defaultValue = "IPA"] attribute phonetic_notation { text }?

[Attributes]

phonetic_notation: Denotes the phonetic alphabet used. Defaults to "IPA". The following values are pre-defined while other user-defined values are allowed as well. However, it is strongly recommended that "IPA" be used when reasonably applicable.

<psp> Denotes a part of speech. Other grammatical marks indicating gender and case may be included here, depending on the policy of the publisher. Its syntax is given in Relax NG compact format below.

psp = element psp { attlist_psp, Inline* }

attlist_psp &= empty

<inflec> Denotes inflection of the word form. Its syntax is given in Relax NG compact format below.

inflec = element inflec { attlist_inflec, Inline* }

attlist_inflec &= empty

<slabel> Denotes category of the terms (e.g. "philosophy", "commerce" and their abbreviations). Its syntax is given in Relax NG compact format below.

slabel = element slabel { attlist_slabel, Inline* }

attlist_slabel &= empty

<guideword> Also denotes the category of the terms. However, they are different from <slabel> in that the latter is mandatory when the (sub)headword falls into one of the applicable categories while the former is optional and intended to be used when they are necessary to distinguish multiple definitions. Its syntax is given in Relax NG compact format below.

guideword = element guideword { attlist_guideword, Inline* }

attlist_guideword &= empty

<spellout> Specifies the repeated string that is to be either replaced with other strings (possibly a symbol) that denotes omission or simply repeated. Its syntax is given in Relax NG compact format below and explained in the following text.

spellout = element spellout { attlist_spellout, Inline* }

attlist_spellout &= attribute org { text }?

[Attributes]

org specifies the string denoting omission. This element specifies how omission should be represented in the final output.

An example is shown in Figure A.70.

<example><spellout org = "～" >spell</spellout> one's name full
</example>

～one's name full

Figure A.70 – Example of the example element with its descendant

<variant> Denotes variants in a Standard character string. Its syntax is given in Relax NG compact format below.

variant = element variant { attlist_variant, Inline* }
attlist_variant &= empty

<etymology> Denotes etymological information in a standard character string. Its syntax is given in Relax NG compact format below.

etymology = element etymology { attlist_etymology, Inline* }
attlist_etymology &= empty

An example of the dict_data element with its descendants is shown in Figure A.71.

```
<dict_data>
<dict_default_attribute>…</dict_default_attribute>
<dict_body>
<dic-item page_break=" yes" turning_page_control="off">
<head>
<headword table_id="ST0001 ">やま【山】</headword><key>やま</key>
<headword table_id="ST0001 ">やま【山】</headword><key>山</key>
</head>
…
<dic-item>
<dic-item …revision=" 2">…<dic-item>
<dic-item …revision=" 1" delete=" yes">…<dic-item>
…
</dict_body>
</dict_data>
```

Figure A.71 – Example of the dict_data element with its descendants

A.4.6.4 Image object instance

Regular image file referred to by the <object> element of the text object instance (see A.4.6.2.3). It can either be a PNG file or a JPEG file.

A.4.6.5 Sound object instance

Regular sound file referred to by the <sound_object_entry> element, or in the <flip_animation> element. MP3, AAC and SMAF files may be used.

A.4.6.6 Animation object instance

The following elements are eligible for animation object instances.

The image sequence animation <flip_animation> defines an animation as a sequence of images changing at fixed time intervals. Written as an XML file, as follows.

The syntax of the <flip_animation> element is given in Relax NG compact format below and explained in the following text.

```
flip_animation =
  element flip_animation {
    attlist_flip_animation,
    flip_animation_sound?,
    flip_animation_source+
  }
attlist_flip_animation &=
  [ a:defaultValue = "1s" ] attribute renewal_time { text }?
```

[Attribute]

renewal_time: Defines the interval between image switches, written in the standard Time format for durations up to 60 s. Defaults to "1s" when omitted. However, some viewers may not be able to respect the defined delay due to capability limitations.

[Child elements]

<flip_animation_sound> Defines a sound object that is to be played simultaneously to the animation. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flip_animation_sound =
  element flip_animation_sound { attlist_flip_animation_sound, empty }
attlist_flip_animation_sound &=
  attribute type { text },
  attribute src { text }
```

[Attributes]

type: Records the MIME type of the sound file. In this annex, only "audio/mp3" is permitted. Shall not be omitted.

src: Name of the sound file, written in the standard filename format. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

<flip_animation_source> Each still image that composes the animation is recorded in separate instances of this element. There shall be at least one instance of this element. When there is more than one of this type of element in the definition of an animation object, all the images referred to by the src attribute shall have the same size. Its syntax is given in Relax NG compact format below and explained in the following text.

```
flip_animation_source =
  element flip_animation_source { attlist_flip_animation_source, empty }
attlist_flip_animation_source &=
  attribute type { text },
  attribute src { text },
  attribute renewal_time { text }?
```

[Attributes]

type: Defines the type of image file registered by the src attribute as a MIME type. Shall not be omitted. The following formats are accepted:

"image/png"

"image/jpeg"

src: Name of the image file, written in the standard Filename format. Shall not be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

renewal_time: Time to wait before switching to the next image, written in the standard time format, with a maximum of 60 s. If omitted, it defaults to the value set in the renewal_time attribute of <flip_animation>.

When the animation object is displayed in a loop (with the loop attribute of <object> set to more than "1"), the loop-time is determined as follows:

If the animation includes a sound file: The animation is regarded as finished when all images have been shown and the associated sound has been played entirely. Repetition starts at this point. Note that the image sequence shall be shown again from the first image.

Otherwise: The animation simply starts again after it reaches the end.

An example is shown in Figure A.72.

```
<flip_animation renewal_time="500ms" >
        <flip_animation_source src="aaa1.jpg" type="image/jpeg"/>
        <flip_animation_source src="aaa2.jpg" type="image/jpeg"/>
        ...
        <flip_animation_source src="aaan.jpg" type="image/jpeg" renewal_time="1s" />
</flip_animation>
```

Figure A.72 – Example of the flip_animation element with its descendants

A.4.6.7 Search page object instance <search_page>

The <search_page> element defines what is rendered on the search page. Its syntax is given in Relax NG compact format below and explained in the following text.

```
search_page =
  element search_page {
    attlist_search_page,
    search_page_title?,
    key_input_region,
    key_input_region?,
    search_link_item*
  }
attlist_search_page &= empty
```

A search page object instance is written as shown in the following example.

An example is shown in Figure A.73.

```
<search_page>
        <search_page_title type="image/png" src="image1.png">
                Foobar dictionary
        </search_page_title>
        <key_input_region table_id="ST0001" search_type="matches_first">
                ...
        </key_input_region>
        <key_input_region table_id="ST0002" search_type="match_only">
                ...
        </key_input_region>
        <search_link_item char_id="CR0001"> ... </search_link_item>
        <search_link_item char_id="CR0002"> ... </search_link_item>
        ...
</search_page>
```

Figure A.73 – Example of the search_page element with its descendants

A sample rendering is shown in Figure A.74. As can be seen, it is divided in 4 regions, as described below. How these regions are visually organized is dependent on the viewer. Some viewers may even decide not to display all of them.

Title string: Region displaying the title of the search page. Defined in the <search_page_title> element.

Title image: Region displaying the search page's title image. Defined in the <search_page_title> element.

Keyword input field: Region to let the user input the keywords on which the search should be based. It is composed of actual input fields, as well as explanatory strings such as "Reading?" or "Spelling?". Each line is associated with a search table, against which the keywords are matched in order to produce the search result. Defined in the <key_input_region> element.

Link list: Displays a list of links to other search pages or specific parts of the documents. Defined in the <search_link_item> element.

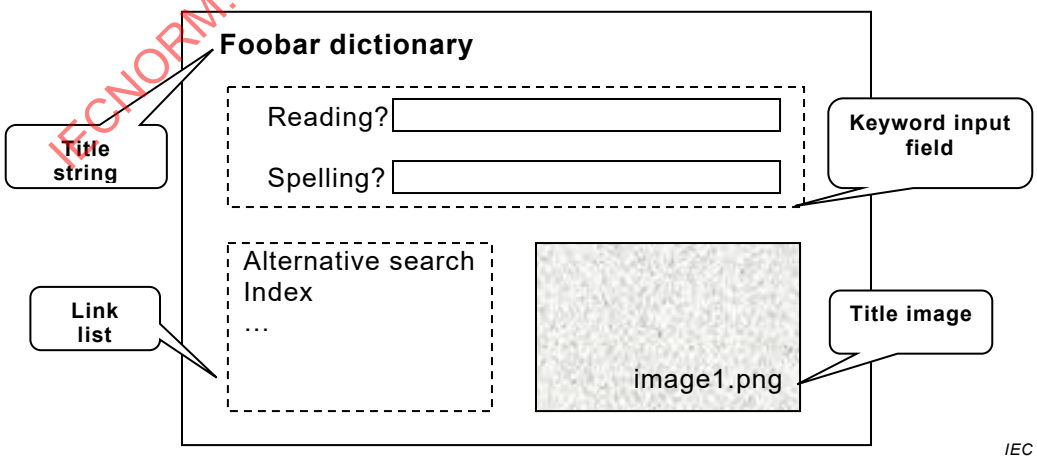


Figure A.74 – Example of search page object instance rendering

[Child elements]

<search_page_title> Defines the title of the search screen. May be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
search_page_title =
  element search_page_title { attlist_search_page_title, TextWithGaiji }
attlist_search_page_title &=
  attribute type { text }?,
  attribute src { text }?
```

[Attributes]

type: Stores the MIME type of the image to use as a title image. May be omitted only if the src attribute is also omitted. The possible values are listed below.

"image/png"

"image/jpeg"

src: Sets the filename of the image to use as a title image, written in the standard filename format. May be omitted. Before opening the file specified in this attribute, it shall be checked against the type attribute.

[Child elements]

The string to be used as a title is written inside the element as an external extended character string.

An example is shown in Figure A.75.

```
<search_page_title type="image/jpeg" src="image.jpg">
        Foobar dictionary
</search_page_title>
```

Figure A.75 – Example of the search_page element

<key_input_region> Defines the keyword input region. Within a single search screen, there can only be one or two instances of this element. The viewer uses this information to display the input region, and then conducts the search based on user input. Its syntax is given in Relax NG compact format below and explained in the following text.

```
key_input_region =
  element key_input_region {
    attlist_key_input_region, key_input_region_prompt, enable_key_type
  }
attlist_key_input_region &=
  attribute table_id { text },
  attribute search_type { "matches_only" | "matches_first" }
```

[Attributes]

table_id: Records the ID number of the search table to be used as a base for the search. It corresponds to the number defined in <search_table_def> (see A.4.3.2.6). Not to be confused with Reference ID (see A.4.6.3.4). Shall not be omitted.

search_type: Defines the type of search, to be chosen from the possibilities listed below. May be omitted, and defaults to "matches_only" in that case.

"matches_only": "Matches only" search mode. The normalized (see A.4.3.2.6, <key_normalization>) keyword is forward-matched to each entry word in the search table designated by the table_id attribute. The result of this search is all the words that match using those criteria, and only those words as as illustrated in Figure A.76.

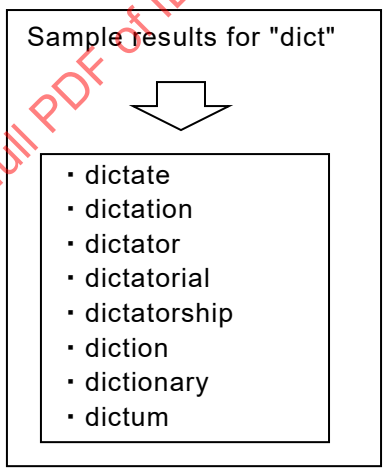


Figure A.76 – Example of search result with search_type set to "matches_only"

"matches_first": "Matches_first" mode. The comparison method is the same as in "matches_only", but the result list is a little different. It contains all the words matching the normalized keyword, and also the following entries in the search table as illustrated in Figure A.77.

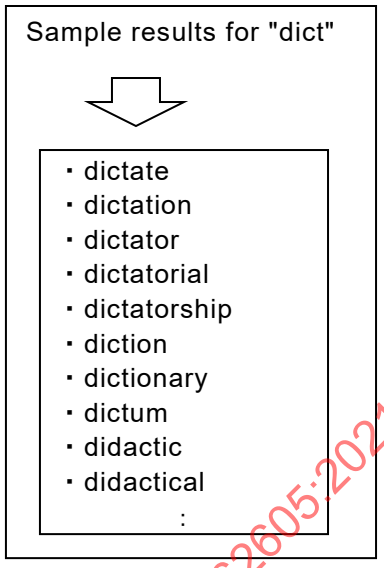


Figure A.77 – Example of search result with search_type set to "matches_first"

[Child elements]

<key_input_region_prompt> Records the character string used as a label to the keyword input field, such as the "Reading?" or "Spelling?" as shown in Figure A.74. Shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
key_input_region_prompt =
  element key_input_region_prompt {
    attlist_key_input_region_prompt, TextWithGaiji
  }
attlist_key_input_region_prompt &= empty
```

[Child elements

The character string to be used as explained above.

<enable_key_type> Defines what kind of character is allowed in the keyword input field. The details of this element are given in A.4.3.2.6. The character types allowed in the search page object's <enable_key_type> shall be a subset of those defined in the associated search table. Shall not be omitted.

Examples are shown in Figure A.78.

```
<key_input_region table_id="ST0001" search_type="matches_first">
        <key_input_region_prompt>Reading?</key_input_ region_prompt>
        <enable_key_type number="no" alphabet="yes" kana="yes"/>
</key_input_region>

<key_input_region table_id="ST0002" search_type="match_only">
        <key_input_region_prompt>Spelling?</key_input_ region_prompt>
        <enable_key_type number="no" alphabet="no" kana="yes"/>
</key_input_region>
```

Figure A.78 – Examples of the key_input_region element

<search_link_item> Records the information defining the link list. They may point to other search tables, or to a specific position in a text flow. The jumps are handled by using the <trigger_pointer> (see A.4.4.3) and <action_page_jump> (see A.4.4.4.3) elements. They should be recorded in the event data (see A.4.4.2) of the flow hosting this search page object. Shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
search_link_item =
  element search_link_item {
    attlist_search_link_item, search_link_title
  }
attlist_search_link_item &= attribute char_id { text }
```

[Attribute]

char_id: Defines the ID number of this search link item, written in the standard char ID format. Shall not be omitted.

[Child elements]

<search_link_title> Sets the title of this link, that is, the text that will be displayed and act as the link. It is written as an External extended character string. Shall not be omitted. Its syntax is given in Relax NG compact format below and explained in the following text.

```
search_link_title =
  element search_link_title { attlist_search_link_title, TextWithGaiji }
attlist_search_link_title &= empty
```

Examples are shown in Figure A.79.

```
<search_link_item char_id="CR0001">
        <search_link_title>Alternate search</search_link_title>
</search_link_item>

<search_link_item char_id="CR0002">
        <search_link_title>Index</search_link_title>
</search_link_item>
```

Figure A.79 – Examples of the search_link_item and search link_title elements

A.4.6.8 Movie object instance

Regular movie file referred to by the <movie_object_entry> element (see A.4.5.5). In this edition of IEC 62605, only 3GPP2 files are allowed.

A.5 Available colour names

The colour names that may be used in the attributes specifying colour types are listed in Table A.6, with the corresponding RGB values.

Table A.6 – Colour names

Colour name	White	Black	Red	Green	Blue
RGB Value	#FFFFFF	#000000	#FF0000	#008000	#0000FF
Colour name	Yellow	Purple	Aqua	Maroon	Navy
RGB Value	#FFFF00	#800080	#00FFFF	#800000	#000080
Colour name	Olive	Teal	Grey	Silver	SlateBlue
RGB Value	#808000	#008080	#808080	#C0C0C0	#6A5ACD
Colour name	MediumBlue	RoyalBlue	DodgerBlue	SkyBlue	SteelBlue
RGB Value	#0000CD	#4169E1	#1E90FF	#87CEEB	#4682B4
Colour name	LightBlue	PaleTurquoise	Turquoise	Cyan	LightCyan
RGB Value	#ADD8E6	#AFEEEE	#40E0D0	#00FFFF	#E0FFFF
Colour name	Aquamarine	DarkGreen	SeaGreen	LightGreen	Chartreuse
RGB Value	#7FFFD4	#006400	#2E8B57	#90EE90	#7FFF00
Colour name	GreenYellow	LimeGreen	YellowGreen	OliveDrab	DarkKhaki
RGB Value	#ADFF2F	#32CD32	#9ACD32	#6B8E23	#BDB76B
Colour name	PaleGoldenrod	LightYellow	Gold	Goldenrod	DarkGoldenrod
RGB Value	#EEE8AA	#FFFFE0	#FFD700	#DAA520	#B8860B
Colour name	RosyBrown	IndianRed	SaddleBrown	Sienna	Peru
RGB Value	#BC8F8F	#CD5C5C	#8B4513	#A0522D	#CD853F
Colour name	Burlywood	Beige	Wheat	SandyBrown	Tan
RGB Value	#DEB887	#F5F5DC	#F5DEB3	#F4A460	#D2B48C
Colour name	Chocolate	Firebrick	Brown	Salmon	Orange
RGB Value	#D2691E	#B22222	#A52A2A	#FA8072	#FFA500
Colour name	Coral	Tomato	HotPink	Pink	DeepPink
RGB Value	#FF7F50	#FF6347	#FF69B4	#FFC0CB	#FF1493
Colour name	PaleVioletRed	Magenta	Violet	Plum	Orchid
RGB Value	#DB7093	#FF00FF	EE82EE	#DDA0DD	#DA70D6
Colour name	DarkViolet	BlueViolet	MediumPurple	Thistle	Lavender
RGB Value	#9400D3	#8A2BE2	#9370DB	#D8BFD8	#E6E6FA
Colour name	MistyRose	Ivory	LemonChiffon	Moccasin	
RGB Value	#FFE4E1	#FFFFF0	#FFFACD	#FFE4B5	

A.6 Localization

A.6.1 Possible additions

The proposals made here for localization are not normative. They are given as examples of additions that should be made to the format before use for a particular language. This localization part is subject to standardization on a language-specific basis when the specification is put to use for that particular language.

A.6.2 Standard characters

For localization purposes, the character sets listed in Table A.7 may be used for the standard character set of the document, depending on the language.

Table A.7 – Examples of additional standard character sets

Target language	Name	Description
Japanese	"JIS X 0201,JIS X 0208:1997"	Characters in the range of Shift_JIS
English	"ISO 646-IRV"	The 7-bit ASCII characters
French	"ISO 8859-15"	Characters in the range of ISO 8859-15 (Latin-9)

A.6.3 Characters usable for reading

In addition to the basic characters allowed in the reading data format, the language-specific extensions listed in Table A.8 are proposed.

Table A.8 – Example of additional characters usable for readings

Target language	Name	Description
Japanese	Full-width Japanese katakana characters	ァアィイゥウェエォオカガキギクグケゲコゴサザシジスズセゼソゾタダチヂッツヅテデトドナニヌネノハバパヒビピフブプヘベペホボポマミムメモャヤュユョヨラリルレロヮワヰヱヲンヴヵヶ (U+30A1 to U+30F6)
Japanese	Japanese long vowel mark	ー (U+30FC)
French	French alphabet extensions	À (U+00C0) Â (U+00C2) Æ (U+00C6) Ç (U+00C7) È (U+00C8) É (U+00C9) Ê (U+00CA) Ë (U+00CB) Î (U+00CE) Ï (U+00CF) Ô (U+00D4) Ö (U+00D6) Ù (U+00D9) Û (U+00DB) Ü (U+00DC) Ÿ (U+00DF) à (U+00E0) â (U+00E2) æ (U+00E6) ç (U+00E7) è (U+00E8) é (U+00E9) ê (U+00EA) ë (U+00EB) î (U+00EE) ï (U+00EF) ô (U+00F4) ù (U+00F9) û (U+00FB) ü (U+00FC) ÿ (U+0FF) Œ(U+0152) œ (U+0153)

A.6.4 Line breaking methods

A.6.4.1 For Japanese

Possible value for the method attribute of the <flow_default_line_breaking_method> (see A.4.3.2.3) and <line_breaking_method> (see A.4.6.2.2) elements.

"run_down": move characters from the end of a line to the beginning of the next one, to respect the position restrictions expressed by the <top_prohibit_char> and <end_prohibit_char> elements.

Additional attributes of the <flow_default_line_breaking_method> and <line_breaking_method> elements when using the "run_down" method.

hanging_punctuation: Activates or deactivates the processing of hanging characters. Possible values are "yes" and "no". Defaults to "yes" when method is set to "run_down". When activated, the characters listed in <hanging_char>, instead of being displayed at the beginning of a line, should be displayed after the end (i.e. in the right margin) of the previous line. Does not apply if the character is the first of a paragraph.

Additional child elements of the <flow_default_line_breaking_method> and <line_breaking_method> elements when using the "run_down" method.

<top_prohibit_char> Lists the characters that shall not appear at the beginning of a line, except as the fist character of the paragraph. Listed as an Extended character string. Defaults to an empty list when omitted.

An example is shown in Figure A.80.

```
<top_prohibit_char>)}]"</top_prohibit_char>
```

Figure A.80 – Example of the top_prohibit_char element

<end_prohibit_char> Lists the characters that shall not appear at the end of a line, except as the last letter of the paragraph. Listed as an extended character string. Defaults to an empty list when omitted.

An example is shown in Figure A.81.

```
<end_prohibit_char>({["</end_prohibit_char>
```

Figure A.81 – Example of the end_prohibit_char element

<hanging_char> Lists the characters to be displayed as hanging punctuation. Listed as an Extended character string. Defaults to an empty list when omitted.

An example is shown in Figure A.82.

```
<hanging_char>、。！？：；<hanging_char>
```

Figure A.82 – Example of the hanging_char element

A.6.4.2 For English

Possible value for the method attribute of the <flow_default_line_breaking_method> and <line_breaking_method> elements.

"word_wrap": Line breaks can only occur on space characters, unless that would cause text to overflow the line.

A.6.5 Sorting rules for <search_table_def>

The methods listed in Table A.9 are proposed.

Table A.9 – Example of additional sorting rules

Target language	Sorting_rule name	Description
Japanese	shift_jis	Characters are ranked according to their code points in the Shift_JIS encoding. Rank for characters not covered by this encoding is unspecified.
English	en	Characters are ranked according to their code points in ASCII encoding. Rank for characters not covered by this encoding is unspecified.
French	fr	Characters are sorted in the following order: A À Â Æ B C Ç E É È Ê Ë F G H I Î Ï J K L M N O Ô Ö Œ P Q R S T U Ù Û Ü V W X Y Ÿ Z The same order is valid for lower case letters too. Rank for other characters is unspecified, and may simply follow the encoding in use, or the Unicode code points.

A.6.6 Additional attributes for <enable_key_type>

It is proposed that <enable_key_type> have the language-specific attributes listed in Table A.10 in addition to those described in A.4.3.2.6. Each accepts 2 values; "yes" and "no". Defaults to "no" when omitted.

Table A.10 – Example of additional language specific attributes for <enable_key_type>

Target language	Character type name	Corresponding characters (all values are in Unicode)
Japanese	fullwidth_numerals	０ to ９ (U+FF10 to U+FF19)
Japanese	fullwidth_alphabet	Ａ to Ｚ (U+FF21 to U+FF3A) and ａ to ｚ (U+FF41to U+FF5A)
Japanese	kana	Half-width katakana: ｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿﾀﾁﾂﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎﾏﾐﾑﾒﾓﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝﾞﾟ (U+FF71 to U+FF9F) and ｦｧｨｩｪｫｬｭｮｯ (U+FF66 to U+FF6F) Full-width katakana: ァアィイゥウェエォオカガキギクグケゲコゴサザシジスズセゼソゾタダチヂッツヅテデトドナニヌネノハバパヒビピフブプヘベペホボポマミムメモャヤュユョヨラリルレロヮワヰヱヲンヴヵヶ(U+30A1 to U+30F6) Full-width hiragana: ぁあぃいぅうぇえぉおかがきぎくぐけげこごさざしじすずせぜそぞただちぢっつづてでとどなにぬねのはばぱひびぴふぶぷへべぺほぼぽまみむめもゃやゅゆょよらりるれろゎわゐゑをん (U+3041 to U+3093) Long vowels: ー (U+30FC, full-width) and ｰ (U+FF70, half-width)
	kanji	To be appropriately defined, e.g. kanji characters included in JIS X 0208:1997 or JIS X 0213:2004
French	French_alphabet_extensions	À (U+00C0) Â (U+00C2) Æ (U+00C6) Ç (U+00C7) È (U+00C8) É (U+00C9) Ê (U+00CA) Ë (U+00CB) Î (U+00CE) Ï (U+00CF) Ô (U+00D4) Ö (U+00D6) Ù (U+00D9) Û (U+00DB) Ü (U+00DC) Ÿ (U+00DF) à (U+00E0) â (U+00E2) æ (U+00E6) ç (U+00E7) è (U+00E8) é (U+00E9) ê (U+00EA) ë (U+00EB) î (U+00EE) ï (U+00EF) ô (U+00F4) ù (U+00F9) û (U+00FB) ü (U+00FC) ÿ (U+0FF) Œ(U+0152) œ (U+0153)

A.6.7 Normalization methods for <key_normalization>

A.6.7.1 For Japanese language

Additional attributes of the <key_normalization> element (see A.4.3.2.6).

cho_on: Japanese long vowel (cho on) conversion method. Possible values are "delete" (remove the character), "repeat" (repeat the vowel preceding the character) and "no" (do nothing). Defaults to "delete".

daku_on: Japanese voiced sound (daku on) conversion. Possible values are "yes" (convert to unvoiced, or sei on) and "no" (no conversion). Defaults to "no".

handaku_on: Japanese semi-voiced sound (handaku on) conversion. Possible values are "yes" (convert to unvoiced) and "no" (no conversion). Defaults to "no".

soku_on: Japanese geminate consonant (soku on or "small tsu") conversion. Possible values are "yes" (convert to large character) and "no" (no conversion). Defaults to "no".

yo_on: Japanese palatalization (yo on) conversion. Possible values are "yes" (conversion to large character) or "no" (no conversion). Defaults to "no".

other_small_kana: Other Japanese small character (hiragana and katakana) conversion. Possible values are "yes" (convert to large character) and "no" (do not convert). Defaults to "no".

In addition to the methods controlled by the preceding parameters, the following normalization process should be applied all the time.

All alphanumeric characters, both full-width and half-width, should be normalized to half-width.

All Japanese hiragana and katakana, both full-width and half-width, should be normalized to full-width katakana.

A.6.7.2 For French language

Additional attributes of the <key_normalization> element.

diactritic_removal: drops all diacritics (accents, tremas, and cedilla) from the letters that carry them. Possible values are "yes" and "no". Defaults to "no".

A.6.8 Character encoding conversions

Converting a piece of text from one character encoding to another can bring some ambiguities, since different encodings, even for the same language, will sometimes cover slightly different character sets. During the localization process such ambiguities should be considered, and the preferred conversion schemes should be specified.

The W3C's XML Japanese profile (http://www.w3.org/TR/japanese-xml/) is a good example of referencing such ambiguities in the Japanese language.

A.7 Adaptation

This annex specifies only the most basic media files as usable. It is to be understood that other media files may be used with media types and other values appropriately defined as needed.

A.8 Specification of the XMDF-LeXML format in the RELAX NG syntax

The syntax of the XMDF-LeXML format is formally given here as a compact RELAX NG schema. Note that it does not include any localization-related addition.

```
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"


  Yes_No = "yes" | "no"
  Trigger_List = trigger_pointer
  Action_List = action_play | action_page_jump
  TextWithGaiji = (text | external_char)*
  FiveSize = "maximum" | "big" | "medium" | "small" | "minimum"
  BaseLine = "right" | "right_only" | "down" | "down_only"
  ViewType = "portrait" | "portrait_only" | "landscape" | "landscape_only"
```

```
HoldFlag = "scope" | "on_power" | "save"
Turn_Page_Val = "on" | "off" | "forward" | "back"
personal_name =
  element personal_name {
    attlist_personal_name,
    ((first_name?, middle_name?, last_name?)
     | (first_name?, last_name?, middle_name?)
     | (last_name?, first_name?, middle_name?)
     | (last_name?, middle_name?, first_name?))
  }
attlist_personal_name &= empty
first_name = element first_name { attlist_first_name, TextWithGaiji }
attlist_first_name &= attribute reading { text }?
middle_name = element middle_name { attlist_middle_name, TextWithGaiji }
attlist_middle_name &= attribute reading { text }?
last_name = element last_name { attlist_last_name, TextWithGaiji }
attlist_last_name &= attribute reading { text }?
organization_name =
  element organization_name { attlist_organization_name, TextWithGaiji }
attlist_organization_name &= attribute reading { text }?
address_info =
  element address_info {
    attlist_address_info,
    postal_code?,
    address?,
    telephone?,
    fax?,
    mail_address?,
    website?,
    address_other_info?
  }
attlist_address_info &= empty
postal_code = element postal_code { attlist_postal_code, text }
attlist_postal_code &= empty
address = element address { attlist_address, TextWithGaiji }
attlist_address &= empty
telephone = element telephone { attlist_telephone, text }
attlist_telephone &= empty
fax = element fax { attlist_fax, text }
attlist_fax &= empty
mail_address = element mail_address { attlist_mail_address, text }
attlist_mail_address &= empty
website = element website { attlist_website, text }
```

```
attlist_website &= empty
address_other_info =
  element address_other_info {
    attlist_address_other_info, TextWithGaiji
  }
attlist_address_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
permission_info =
  element permission_info {
    attlist_permission_info, print_permission?, copy_permission?
  }
attlist_permission_info &= empty
print_permission =
  element print_permission { attlist_print_permission, empty }
attlist_print_permission &=
  [ a:defaultValue = "no" ]
  attribute permission { "authorized" | "no" }?
copy_permission =
  element copy_permission { attlist_copy_permission, empty }
attlist_copy_permission &=
  [ a:defaultValue = "no" ]
  attribute permission { "authorized" | "in_device_only" | "no" }?
keyword_list = element keyword_list { attlist_keyword_list, keyword+ }
attlist_keyword_list &= empty
keyword = element keyword { attlist_keyword, TextWithGaiji }
attlist_keyword &=
  attribute category { text }?,
  attribute reading { text }?
vertex = element vertex { attlist_vertex, empty }
attlist_vertex &= attribute position { text }
bvf = element bvf { attlist_bvf, book_info, body_module, parts_module }
attlist_bvf &=
  attribute id_type { text }?,
  attribute id { text }?,
  attribute default_ccs { text },
  attribute display_size { text }?
book_info =
  element book_info {
    attlist_book_info,
    title_info,
    author_info?,
    publisher_info?,
```

```
    seller_info?,
    book_id_info?,
    classification_info?,
    rating?,
    publication_place?,
    publication_date_info?,
    net_price_info?,
    book_abstract?,
    front_cover_image?,
    spine_cover_image?,
    keyword_list?,
    other_book_info?
  }
attlist_book_info &= empty
title_info =
  element title_info {
    attlist_title_info,
    series_title?,
    title,
    subtitle?,
    edition_info?,
    title_other_info?
  }
attlist_title_info &= empty
series_title =
  element series_title { attlist_series_title, TextWithGaiji }
attlist_series_title &= attribute reading { text }?
title = element title { attlist_title, TextWithGaiji }
attlist_title &= attribute reading { text }?
subtitle = element subtitle { attlist_subtitle, TextWithGaiji }
attlist_subtitle &= attribute reading { text }?
edition_info =
  element edition_info { attlist_edition_info, TextWithGaiji }
attlist_edition_info &=
  attribute this_version { text }?,
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
title_other_info =
  element title_other_info { attlist_title_other_info, TextWithGaiji }
attlist_title_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
author_info = element author_info { attlist_author_info, author+ }
```

```
attlist_author_info &= empty
author =
  element author {
    attlist_author,
    (personal_name | organization_name),
    address_info?,
    author_other_info?
  }
attlist_author &=
  [ a:defaultValue = "author" ]
  attribute role {
    "author"
    | "editor"
    | "translator"
    | "supervisor"
    | "designer"
    | "photographer"
    | "illustrator"
    | "binder"
    | "planner"
    | "other"
  }?
author_other_info =
  element author_other_info { attlist_author_other_info, TextWithGaiji }
attlist_author_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
publisher_info =
  element publisher_info {
    attlist_publisher_info,
    ((publisher, publisher_office) | publisher | publisher_office),
    publisher_other_info?
  }
attlist_publisher_info &= empty
publisher =
  element publisher { attlist_publisher, publisher_name, address_info? }
attlist_publisher &= empty
publisher_name =
  element publisher_name { attlist_publisher_name, TextWithGaiji }
attlist_publisher_name &= attribute reading { text }?
publisher_office =
  element publisher_office {
    attlist_publisher_office, organization_name, address_info?
```

```
  }
attlist_publisher_office &= attribute publisher_code { text }?
publisher_other_info =
  element publisher_other_info {
    attlist_publisher_other_info, TextWithGaiji
  }
attlist_publisher_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
seller_info =
  element seller_info {
    attlist_seller_info,
    ((seller, seller_office) | seller | seller_office),
    seller_other_info?
  }
attlist_seller_info &= empty
seller = element seller { attlist_seller, seller_name, address_info? }
attlist_seller &= empty
seller_name = element seller_name { attlist_seller_name, TextWithGaiji }
attlist_seller_name &= attribute reading { text }?
seller_office =
  element seller_office {
    attlist_seller_office, organization_name, address_info?
  }
attlist_seller_office &= attribute seller_code { text }?
seller_other_info =
  element seller_other_info { attlist_seller_other_info, TextWithGaiji }
attlist_seller_other_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
book_id_info = element book_id_info { attlist_book_id_info, book_id+ }
attlist_book_id_info &= empty
book_id = element book_id { attlist_book_id, text }
attlist_book_id &= attribute type { text }
classification_info =
  element classification_info {
    attlist_classification_info, classification+
  }
attlist_classification_info &= empty
classification =
  element classification { attlist_classification, TextWithGaiji }
attlist_classification &= attribute type { text }
rating = element rating { attlist_rating, empty }
```

```
attlist_rating &=
  [ a:defaultValue = "no" ] attribute adult { Yes_No }?,
  [ a:defaultValue = "no" ] attribute violence { Yes_No }?
publication_place =
  element publication_place { attlist_publication_place, text }
attlist_publication_place &= empty
publication_date_info =
  element publication_date_info {
    attlist_publication_date_info, publication_date+
  }
attlist_publication_date_info &= empty
publication_date =
  element publication_date { attlist_publication_date, text }
attlist_publication_date &=
  [ a:defaultValue = "publish" ] attribute type { "publish" | "sale" }?
net_price_info =
  element net_price_info { attlist_net_price_info, net_price+ }
attlist_net_price_info &= empty
net_price = element net_price { attlist_net_price, text }
attlist_net_price &=
  attribute country { text }?,
  attribute unit { text },
  attribute other_info { text }?
book_abstract =
  element book_abstract { attlist_book_abstract, TextWithGaiji }
attlist_book_abstract &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
front_cover_image =
  element front_cover_image { attlist_front_cover_image, text }
attlist_front_cover_image &= attribute type { text }
spine_cover_image =
  element spine_cover_image { attlist_spine_cover_image, text }
attlist_spine_cover_image &= attribute type { text }
other_book_info =
  element other_book_info { attlist_other_book_info, TextWithGaiji }
attlist_other_book_info &=
  [ a:defaultValue = "preserve" ]
  attribute xml:space { "default" | "preserve" }?
body_module =
  element body_module { attlist_body_module, flow_type_body }
attlist_body_module &= empty
flow_type_body =
```

```
  element flow_type_body {
    attlist_flow_type_body,
    flow_entry,
    special_page_link?,
    search_table?
  }
attlist_flow_type_body &= empty
flow_entry =
  element flow_entry {
    attlist_flow_entry, flow_default_attribute?, flow_data+
  }
attlist_flow_entry &= empty
flow_default_attribute =
  element flow_default_attribute {
    attlist_flow_default_attribute,
    flow_default_size?,
    flow_default_font?,
    flow_default_background?,
    flow_default_line_breaking_method?
  }
attlist_flow_default_attribute &=
  attribute baseline { BaseLine }?,
  attribute view_type { ViewType }?,
  [ a:defaultValue = "IPA" ] attribute phonetic_notation { text }?
flow_default_size =
  element flow_default_size { attlist_flow_default_size, empty }
attlist_flow_default_size &=
  attribute letter_spacing { FiveSize }?,
  attribute line_pitch { FiveSize }?,
  attribute margin { "big" | "medium" | "small" }?
flow_default_font =
  element flow_default_font { attlist_flow_default_font, empty }
attlist_flow_default_font &=
  attribute fontname { text }?,
  attribute fontsize { FiveSize }?,
  attribute bold_flag { Yes_No }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute colour { text }?,
  attribute ruby_flag { "yes" | "yes_only" | "no" | "no_only" }?,
  attribute italic { Yes_No }?,
  attribute oblique { Yes_No }?,
  attribute small_caps { Yes_No }?,
```

```
  attribute family {
    "monospace" | "san-serif" | "serif" | "cursive" | "fantasy"
  }?,
  attribute ul_type { "rightscore" | "leftscore" | "throughscore" }?,
  attribute em_type {
    "rightscore"
    | "leftscore"
    | "throughscore"
    | "kendot"
    | "bold"
    | "italic"
    | "bold_italic"
    | "reverse"
    | "shade"
  }?
flow_default_background =
  element flow_default_background {
    attlist_flow_default_background, empty
  }
attlist_flow_default_background &=
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?
flow_default_line_breaking_method =
  element flow_default_line_breaking_method {
    attlist_flow_default_line_breaking_method, empty
  }
attlist_flow_default_line_breaking_method &=
  [ a:defaultValue = "none" ] attribute method { "none" }?
flow_default_delimiter =
  element flow_default_delimiter {
    attlist_flow_default_delimiter, delimiter?
  }
attlist_flow_default_delimiter &= empty
delimiter = element delimiter { attlist_delimiter, empty }
attlist_delimiter &=
  attribute tag { text },
  attribute type { text }?,
  [ a:defaultValue = "" ] attribute open { text }?,
  [ a:defaultValue = "" ] attribute close { text }?
flow_data = element flow_data { attlist_flow_data, event_info? }
attlist_flow_data &=
  attribute flow_id { text }?,
```

```
    attribute body_id { text },
    [ a:defaultValue = "off" ]
    attribute turning_page_control { Turn_Page_Val }?
  special_page_link =
    element special_page_link { attlist_special_page_link, special_page+ }
  attlist_special_page_link &= empty
  special_page = element special_page { attlist_special_page, text }
  attlist_special_page &=
    [ a:defaultValue = "other" ]
    attribute kind {
      "cover"
      | "title_page"
      | "preface"
      | "contents"
      | "body"
      | "column"
      | "note"
      | "figure"
      | "ad"
      | "afterword"
      | "appendix"
      | "answer"
      | "glossary"
      | "bibliography"
      | "commentary"
      | "index"
      | "imprint"
      | "author_info"
      | "other"
    }?,
    attribute title { text }?
  search_table =
    element search_table { attlist_search_table, search_table_def+ }
  attlist_search_table &=
    [ a:defaultValue = "no" ] attribute bookmark { Yes_No }?,
    [ a:defaultValue = "no" ] attribute wordbook { Yes_No }?,
    [ a:defaultValue = "no" ] attribute jump_search_root { Yes_No }?,
    [ a:defaultValue = "no" ] attribute jump_search { Yes_No }?,
    [ a:defaultValue = "no" ] attribute all_search { Yes_No }?
  search_table_def =
    element search_table_def {
      attlist_search_table_def, enable_key_type, key_normalization
    }
```

```
attlist_search_table_def &=
  attribute id { text },
  [ a:defaultValue = "no" ] attribute use_default { Yes_No }?,
  [ a:defaultValue = "unicode" ] attribute sorting_rule { "unicode" }?,
  attribute name { text },
  attribute short_name { text },
  attribute wild { Yes_No },
  attribute blank { Yes_No },
  attribute end { Yes_No },
  attribute help_page_id { text }?
enable_key_type =
  element enable_key_type { attlist_enable_key_type, empty }
attlist_enable_key_type &=
  [ a:defaultValue = "no" ] attribute numerals { Yes_No }?,
  [ a:defaultValue = "no" ] attribute basic_alphabet { Yes_No }?
key_normalization =
  element key_normalization { attlist_key_normalization, empty }
attlist_key_normalization &=
  [ a:defaultValue = "yes" ] attribute capitalization { Yes_No }?
event_info = element event_info { attlist_event_info, event+ }
attlist_event_info &=
  [ a:defaultValue = "single" ] attribute display_type { "single" }?
event = element event { attlist_event, trigger, action }
attlist_event &= empty
trigger = element trigger { attlist_trigger, Trigger_List }
attlist_trigger &= empty
action = element action { attlist_action, Action_List }
attlist_action &=
  [ a:defaultValue = "sequential" ]
  attribute action_flag { "sequential" }?
trigger_pointer =
  element trigger_pointer { attlist_trigger_pointer, pointer_region? }
attlist_trigger_pointer &=
  attribute id { text },
  [ a:defaultValue = "click" ] attribute action_flag { "click" }?
pointer_region =
  element pointer_region { attlist_pointer_region, vertex+ }
attlist_pointer_region &= empty
action_play = element action_play { attlist_action_play, empty }
attlist_action_play &=
  attribute object_id { text },
  [ a:defaultValue = "normal" ] attribute action { "normal" }?
action_page_jump =
```

```
  element action_page_jump { attlist_action_page_jump, empty }
attlist_action_page_jump &=
  attribute book { text }?,
  attribute book_type { text }?,
  attribute page_id { text }?,
  attribute center { text }?
parts_module =
  element parts_module { attlist_parts_module, object_table }
attlist_parts_module &= empty
object_table =
  element object_table {
    attlist_object_table,
    (dynamic_text_object_entry
     | sound_object_entry
     | search_page_object_entry
     | movie_object_entry
     | dict_data_object_entry)+
  }
attlist_object_table &= empty
dynamic_text_object_entry =
  element dynamic_text_object_entry {
    attlist_dynamic_text_object_entry, permission_info?
  }
attlist_dynamic_text_object_entry &=
  attribute src { text },
  attribute type { text },
  attribute object_id { text }
sound_object_entry =
  element sound_object_entry {
    attlist_sound_object_entry, sound_object_info
  }
attlist_sound_object_entry &=
  attribute src { text },
  attribute type { text }
sound_object_info =
  element sound_object_info { attlist_sound_object_info, empty }
attlist_sound_object_info &= attribute object_id { text }
search_page_object_entry =
  element search_page_object_entry {
    attlist_search_page_object_entry, permission_info?
  }
attlist_search_page_object_entry &=
  attribute src { text },
```

```
  attribute type { text },
  attribute object_id { text }
movie_object_entry =
  element movie_object_entry { attlist_movie_object_entry, empty }
attlist_movie_object_entry &=
  attribute src { text },
  attribute type { text },
  attribute object_id { text }
dict_data_object_entry =
  element dict_data_object_entry {
    attlist_dict_data_object_entry, permission_info?
  }
attlist_dict_data_object_entry &=
  attribute src { text },
  attribute type { text },
  attribute object_id { text }
text_data =
  element text_data {
    attlist_text_data, text_default_attribute, text_body
  }
attlist_text_data &= empty
text_default_attribute =
  element text_default_attribute {
    attlist_text_default_attribute,
    text_default_font?,
    text_default_background?,
    text_default_background_music?,
    line_breaking_method?
  }
attlist_text_default_attribute &=
  attribute baseline { BaseLine }?,
  attribute valign { "middle" }?
text_default_font =
  element text_default_font { attlist_text_default_font, empty }
attlist_text_default_font &=
  attribute fontname { text }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute ruby_flag { "yes" | "yes_only" | "no" | "no_only" }?
text_default_background =
  element text_default_background {
    attlist_text_default_background, permission_info?
```

```
  }
attlist_text_default_background &=
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute type { text }?,
  attribute src { text }?
text_default_background_music =
  element text_default_background_music {
    attlist_text_default_background_music, permission_info?
  }
attlist_text_default_background_music &=
  attribute type { text },
  attribute src { text },
  [ a:defaultValue = "no" ] attribute loop { Yes_No }?
line_breaking_method =
  element line_breaking_method { attlist_line_breaking_method, empty }
attlist_line_breaking_method &=
  [ a:defaultValue = "none" ] attribute method { "none" }?
Inline =
  text
  | br
  | hr
  | font
  | horizontal
  | ruby
  | external_char
  | mask
  | char_id
  | i
  | u
  | em
  | oblq
  | sc
  | mlg
  | sub
  | sup
  | meaning
  | example
  | column
  | \div
  | span
  | table
```

```
  | select
  | tts
  | a
  | object
All_tag_p = Inline | p | scrolling_text
All_tag_di = All_tag_p | head | subhead
All_tag = All_tag_di | page_break
text_body = element text_body { attlist_text_body, All_tag* }
attlist_text_body &= empty
p = element p { attlist_p, All_tag_p* }
attlist_p &=
  attribute top_line_indent { text }?,
  attribute top { text }?,
  attribute bottom { text }?,
  attribute align { "center" | "right" | "left" }?,
  attribute drop_cap { text }?
scrolling_text =
  element scrolling_text {
    attlist_scrolling_text,
    (text
     | external_char
     | font
     | horizontal
     | ruby
     | sub
     | sup
     | object)*
  }
attlist_scrolling_text &= empty
page_break = element page_break { attlist_page_break, empty }
attlist_page_break &= attribute turning_page_control { Turn_Page_Val }?
br = element br { attlist_br, empty }
attlist_br &= attribute clear { "left" | "right" | "all" }?
hr = element hr { attlist_hr, empty }
attlist_hr &=
  attribute size { text }?,
  attribute length { text }?,
  attribute align { "left" | "center" | "right" }?
font = element font { attlist_font, Inline* }
attlist_font &=
  attribute name { text }?,
  attribute size { text }?,
  attribute base { "default" | "last" }?,
```

```
    attribute color_space { "RGB" }?,
    attribute opacity { "100" }?,
    attribute color { text }?,
    attribute bold { Yes_No }?,
    attribute underline { Yes_No }?,
    attribute italic { Yes_No }?,
    attribute oblique { Yes_No }?,
    attribute small_caps { Yes_No }?,
    attribute family {
      "monospace" | "san-serif" | "serif" | "cursive" | "fantasy"
    }?,
    attribute ul_type { "rightscore" | "leftscore" | "throughscore" }?,
    attribute em_type {
      "rightscore"
      | "leftscore"
      | "throughscore"
      | "kendot"
      | "bold"
      | "italic"
      | "bold_italic"
      | "reverse"
      | "shade"
    }?
  i = element i { attlist_i, Inline* }
  attlist_i &= empty
  u = element u { attlist_u, Inline* }
  attlist_u &=
    attribute ul_type { "rightscore" | "leftscore" | "throughscore" }?
  em = element em { attlist_em, Inline* }
  attlist_em &=
    attribute em_type {
      "rightscore"
      | "leftscore"
      | "throughscore"
      | "kendot"
      | "bold"
      | "italic"
      | "bold_italic"
      | "reverse"
      | "shade"
    }?
  oblq = element oblq { attlist_oblq, Inline* }
  attlist_oblq &= empty
```

```
sc = element sc { attlist_sc, Inline* }
attlist_sc &= empty
mlg = element mlg { attlist_mlg, Inline* }
attlist_mlg &= empty
horizontal = element horizontal { attlist_horizontal, TextWithGaiji }
attlist_horizontal &= empty
ruby = element ruby { attlist_ruby, rbase, rtop }
attlist_ruby &= empty
rb = element rb { attlist_rb, TextWithGaiji }
attlist_rb &= empty
rt = element rt { attlist_rt, TextWithGaiji }
attlist_rt &= empty
sub = element sub { attlist_sub, TextWithGaiji }
attlist_sub &= empty
sup = element sup { attlist_sup, TextWithGaiji }
attlist_sup &= empty
external_char = element external_char { attlist_external_char, text }
attlist_external_char &=
  attribute alt_set { text }?,
  attribute alt_code { text }?,
  attribute alt_img { text }?,
  attribute alt_vimg { text }?,
  attribute img_type { text }?,
  attribute alt { text }?
mask =
  element mask {
    attlist_mask,
    (text | br | font | horizontal | ruby | external_char | object)*
  }
attlist_mask &=
  [ a:defaultValue = "on" ] attribute initial_flag { "on" | "off" }?,
  attribute trigger { text }?,
  attribute char_id { text }?,
  [ a:defaultValue = "default" ]
  attribute mask_type { "default" | "color" }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  [ a:defaultValue = "black" ] attribute color { text }?,
  [ a:defaultValue = "scope" ] attribute hold_flag { HoldFlag }?
char_id = element char_id { attlist_char_id, Inline* }
attlist_char_id &= attribute char_id { text }
head = element head { attlist_head, headword+, key* }
attlist_head &= empty
```

```
headword = element headword { attlist_headword, TextWithGaiji }
attlist_headword &=
  attribute type { text }?,
  attribute table_id { text },
  [ a:defaultValue = "IPA" ] attribute phonetic_notation { text }?
key = element key { attlist_key, text }
attlist_key &= attribute type { text }?
meaning = element meaning { attlist_meaning, All_tag_p* }
attlist_meaning &=
  attribute type { text }?,
  attribute subid { text }?,
  attribute level { text }?,
  attribute no { text }?
example = element example { attlist_example, All_tag_p* }
attlist_example &= empty
subhead =
  element subhead {
    attlist_subhead, subheadword+, meaning*, example*, key*
  }
attlist_subhead &=
  attribute type { text }?,
  attribute subid { text }?
subheadword = element subheadword { attlist_subheadword, TextWithGaiji }
attlist_subheadword &=
  attribute type { text }?,
  attribute subid { text }?
ref = element ref { attlist_ref, All_tag_p* }
attlist_ref &=
  attribute type { text }?,
  attribute refid { text }
split = element split { attlist_split, All_tag_p* }
attlist_split &= attribute level { text }?
column = element column { attlist_column, All_tag_p* }
attlist_column &=
  attribute type { text }?,
  attribute subid { text }
object = element object { attlist_object, permission_info? }
attlist_object &=
  attribute type { text },
  attribute src { text },
  attribute char_id { text }?,
  attribute align { "top" | "middle" | "bottom" | "left" | "right" }?,
  attribute start { "auto" | "event" }?,
```

```
  attribute loop { "1" }?
\div = element div { attlist_div, All_tag_p* }
attlist_div &=
  attribute level { text }?,
  attribute title { text }?
span = element span { attlist_span, All_tag_p* }
attlist_span &= attribute type { text }?
table = element table { attlist_table, tr+ }
attlist_table &= empty
tr = element tr { attlist_tr, td+, th* }
attlist_tr &= empty
th = element th { attlist_th, Inline* }
attlist_th &= empty
td = element td { attlist_td, Inline* }
attlist_td &= empty
select = element select { attlist_select, select_item+ }
attlist_select &=
  attribute default { text }?,
  attribute type { text }?
select_item = element select_item { attlist_select_item, All_tag_p* }
attlist_select_item &= empty
tts = element tts { attlist_tts, All_tag_p* }
attlist_tts &= attribute speaker { "male" | "female" | "child" }?
a = element a { attlist_a, Inline* }
attlist_a &= attribute href { text }
dict_data =
  element dict_data {
    attlist_dict_data, dict_default_attribute, dict_body
  }
attlist_dict_data &= empty
dict_default_attribute =
  element dict_default_attribute {
    attlist_dict_default_attribute,
    dict_default_font?,
    dict_default_background?,
    dict_default_background_music?,
    line_breaking_method?
  }
attlist_dict_default_attribute &=
  attribute baseline { BaseLine }?,
  attribute valign { "middle" }?
dict_default_font =
  element dict_default_font { attlist_dict_default_font, empty }
```

```
attlist_dict_default_font &=
  attribute fontname { text }?,
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute ruby_flag { "yes" | "yes_only" | "no" | "no_only" }?
dict_default_background =
  element dict_default_background {
    attlist_dict_default_background, permission_info?
  }
attlist_dict_default_background &=
  attribute color_space { "RGB" }?,
  attribute opacity { "100" }?,
  attribute color { text }?,
  attribute type { text }?,
  attribute src { text }?
dict_default_background_music =
  element dict_default_background_music {
    attlist_dict_default_background_music, permission_info?
  }
attlist_dict_default_background_music &=
  attribute type { text },
  attribute src { text },
  [ a:defaultValue = "no" ] attribute loop { Yes_No }?
dict_body = element dict_body { attlist_dict_body, split*, dic-item+ }
attlist_dict_body &= empty
dic-item =
  element dic-item {
    attlist_dic-item,
    (All_tag_di
     | gender
     | psp
     | glabel
     | pronunciation
     | inflec
     | slabel
     | guideword
     | spellout
     | variant
     | etymology)*
  }
attlist_dic-item &=
  attribute type { text }?,
```

```
  attribute id { text },
  attribute rank { text }?,
  attribute level { text }?,
  [ a:defaultValue = "no" ] attribute page_break { Yes_No }?,
  attribute turning_page_control { Turn_Page_Val }?,
  attribute revision { text }?,
  attribute delete { Yes_No }?
gender = element gender { attlist_gender, Inline* }
attlist_gender &= empty
psp = element psp { attlist_psp, Inline* }
attlist_psp &= empty
glabel = element glabel { attlist_glabel, Inline* }
attlist_glabel &= empty
pronunciation = element pronunciation { attlist_pronunciation, Inline* }
attlist_pronunciation &=
  [ a:defaultValue = "IPA" ] attribute phonetic_notation { text }?
inflec = element inflec { attlist_inflec, Inline* }
attlist_inflec &= empty
slabel = element slabel { attlist_slabel, Inline* }
attlist_slabel &= empty
guideword = element guideword { attlist_guideword, Inline* }
attlist_guideword &= empty
spellout = element spellout { attlist_spellout, Inline* }
attlist_spellout &= attribute org { text }?
variant = element variant { attlist_variant, Inline* }
attlist_variant &= empty
etymology = element etymology { attlist_etymology, Inline* }
attlist_etymology &= empty
flip_animation =
  element flip_animation {
    attlist_flip_animation,
    flip_animation_sound?,
    flip_animation_source+
  }
attlist_flip_animation &=
  [ a:defaultValue = "1s" ] attribute renewal_time { text }?
flip_animation_sound =
  element flip_animation_sound { attlist_flip_animation_sound, empty }
attlist_flip_animation_sound &=
  attribute type { text },
  attribute src { text }
flip_animation_source =
  element flip_animation_source { attlist_flip_animation_source, empty }
```

```
attlist_flip_animation_source &=
  attribute type { text },
  attribute src { text },
  attribute renewal_time { text }?
search_page =
  element search_page {
    attlist_search_page,
    search_page_title?,
    key_input_region,
    key_input_region?,
    search_link_item*
  }
attlist_search_page &= empty
search_page_title =
  element search_page_title { attlist_search_page_title, TextWithGaiji }
attlist_search_page_title &=
  attribute type { text }?,
  attribute src { text }?
key_input_region =
  element key_input_region {
    attlist_key_input_region, key_input_region_prompt, enable_key_type
  }
attlist_key_input_region &=
  attribute table_id { text },
  attribute search_type { "matches_only" | "matches_first" }?
key_input_region_prompt =
  element key_input_region_prompt {
    attlist_key_input_region_prompt, TextWithGaiji
  }
attlist_key_input_region_prompt &= empty
search_link_item =
  element search_link_item {
    attlist_search_link_item, search_link_title
  }
attlist_search_link_item &= attribute char_id { text }
search_link_title =
  element search_link_title { attlist_search_link_title, TextWithGaiji }
attlist_search_link_title &= empty
rbase |= notAllowed
rtop |= notAllowed
start =
  dict_data
  | search_page
```

| flip_animation
| ref
| bvf
| text_data
| rt
| rb
| flow_default_delimiter

Annex B
(normative)

LeXML format

B.1 General

The LeXML (LEXicographical eXtensible Markup Language) is a format which aims for structural description of dictionaries, glossaries and encyclopaedias. The format is defined and has been extended on the input from the actual work to represent these contents in XML.

The format is capable of representing monolingual, bilingual and multilingual dictionaries of various sizes, ranging from "pocket" to unabridged dictionaries.

B.2 Elements for content structure

B.2.1 Parameter entity definition

B.2.1.1 %inline.html

LeXML elements consist of two categories. The first category shows HTML-like elements as presented in the (document type definition) DTD below.

NOTE For definitions that appear in the Relax NG compact representation of each element in this annex, see Clause B.4.

```
<!ENTITY % inline.html "
    b |
    br |
    em |
    i |
    nobr |
    span |
    sub |
    sup |
    u |
    ruby |
    a |
    img |
    big |
    small
">
```

B.2.1.2 %inline.lexml

The second category shows the elements defined by LeXML, as listed in the DTD below.

```
<!ENTITY % inline.lexml "
    alabel | glabel | slabel | pos | POS | hinshi | gender | GENDER | pron |
    hatsuon | svoc | ref | REF | xref | XREF | inflec | lang | spellout |
    variant | hidden | light | sc | oblq | audio | video | note | accent |
    ex | etym | ymd | article | ud | hs | pha | ipa | pinyin | cn | jp | kr |
    tw | ggk | kigo | mlg | gi | gix | fbox | FBOX | pro-n | pro-v | abbr |
    fullform | vnum | primary | kanbun ">
```

B.2.2 Root elements

B.2.2.1 dic-body

The root element.

NOTE In pratical applications, it is sometimes preferred to treat data without the root element, i.e. sequence of dic-item elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT dic-body (split | dic-item)*>
```

[Attributes]

None.

B.2.2.2 split

Specifies the delimiter between lexicographical blocks (e.g. this element may be used between the last word starting with "A" and the first word starting with "B"). This element is intended to record the starting points of such blocks as well as what strings should be rendered for that (e.g. "-B-" for a block consisting of words starting with "B".)

The syntax of the element is shown in the DTD below.

```
<!ELEMENT split (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST split type CDATA #IMPLIED>
```

[Attributes]

type

(optional) Specifies the type.

Examples are shown in Figure B.1.

```
<split>A</split>

<dic-item id="A0000100" rank="01">
<head><headword>a, A</headword>....</head>....</dic-item>
<dic-item id="A0000200" rank="compound">
<head><headword>&diams;A battery</headword>....</head>....</dic-item>
. . . . . .
<dic-item id="A0406100" rank="04">
<head><headword>az・ure</headword>....</head>....</dic-item>

<split>B</split>
```

Figure B.1 – Examples of the split element

NOTE XML representation examples may include entities for characters that are not XML standard in this annex. Their values are not given in this document as they are not essential for understanding the format and such entities are outside the scope of this document.

B.2.2.3 dic-item

Defines the logical structure of the entry. This is used as a unit to be used for search and display. Cannot be omitted.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT dic-item (head, (
      meaning | meaning-group | example | example-group | subhead | subheadword |
      index | key | column | div | p | image | audio | video | table | replace |
      ul | dl | memo | data | snippet)+)>
<!ATTLIST dic-item id      ID    #REQUIRED
                  type    CDATA #IMPLIED
                  rank    CDATA #IMPLIED
                  level   CDATA #IMPLIED
                  orgid   CDATA #IMPLIED
                  pid     CDATA #IMPLIED
                  sortkey CDATA #IMPLIED >
```

[Attributes]

id: Specifies an ID of the logical structure (item). Cannot be omitted.

type: (optional) Specifies the type for the logical structure (item).

Attribute value examples: "parent_headword", "kanji_representation".

rank: (optional) Specifies the category such as "important word" for the logical structure (item).

level: (optional) Specifies the information of the hierarchical position/level (in semantics) of the logical structure (item).

Attribute value examples: "14_kanagawa", "14100_yokohama_kanagawa", "1410_tsurumi-ward_yokohama_kanagawa".

orgid: (optional) Specifies the original ID when renumbered, such as the ID in the older version and the ID for management of the source data.

pid: (optional) Specifies an ID of apparent entry (Note that it is given as a sibling dic-item element) when this element is a child entry of another headword.

sortkey:(optional) Specifies the key for sorting the entries.

An example is shown in Figure B.2.

(paper-form)

plasma /plǽzmə/ [noun] [U] **1** 〔ANATOMICAL〕 =blood ~ **2** whey; milk serum **3** 〔BIOLOGICAL〕 protoplasm **4** 〔PHYSICAL〕 plasma; ionized gas

```
<dic-item id="P0287900#0000000" rank="04">
<head>
<headword>plasma</headword>
        <key type="04">plasma</key>
<headword
type="pronunciation">pl&aeacute;zm&schwa;</headword>
</head>
<meaning
subid="P0287900#NN00000"><pos>noun</pos><pos>U</pos></
meaning>
<meaning subid="P0287900#NN01000" level="1"
no="01"><b>1</b> <slabel>ANATOMICAL</slabel> =<ref
refid="B0247800#0000000">blood <spellout org="~
">plasma</spellout></ref></meaning>
<meaning subid="P0287900#NN02000" level="1"
```

Figure B.2 – Example of the dic-item element with its descendants

NOTE Some examples of XML in this annex include visual representations preceding or succeeding a downward arrow like the one above. Those preceding the downward arrow are for paper-form (not necessarily having corresponding XML data), while those succeeding the downward arrow are for digital representation. This reflects the situation where a large volume of XML data has been produced through conversion from paper-form contents.

Another example is shown in Figure B.3.

```
<dic-item id="ABCD00000100">
<head>
<headword>みだし・ご</headword>
<key>みだしご</key>
<headword type="表記">見出し語</headword>
      <key type="表記">見出し語</key>
</head>
<meaning>語義語釈、解説など</meaning>
<example>用例</example>
<subheadword type="子見出し">子[小]見出し（派生語、複合語、成句
など）</subheadword>
      <key type="子見出しかな">こみだし</key>
        <key type="子見出し表記">子見出し</key>
        <key type="子見出し表記">小見出し</key>
</dic-item>
```

Figure B.3 – Example of the dic-item element with its descendants

B.2.3 headword-related elements

B.2.3.1 head

Denotes a block consisting of the headword and other data that forms a set with it (representation, pronunciation, search key, etc.). Cannot be omitted.

The syntax of the element is shown in the DTD below.

<!ELEMENT head (headword | key)*>

[Attributes]

None.

An example is shown in Figure B.4.

pyr·a·mid /pír ə mìd/

```
<head>
<headword>pyr・a・mid</headword>
<key>pyramid</key>
<headword type="発音">p&iacute;r&schwa;m&igrave;d</headword>
```

Figure B.4 – Example of the head element

Another example is shown in Figure B.5

エジプト Egypt【埃及】

```
<head>
<headword>エジプト</headword>
<key>エジプト</key>
<headword type="原綴">Egypt</headword>
<key type="原綴">Egypt</key>
<headword type="漢字">埃及</headword>
```

Figure B.5 – Example of the head element

B.2.3.2 headword

Specifies the headword and other data that form a set with it. Cannot be omitted.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT headword (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST headword type      CDATA #IMPLIED
                   delimiter CDATA #IMPLIED >
```

[Attributes]

type

(optional) Used to distinguish the main headwords (usually do not have this attribute) from other headwords. For example, being a variant representation of the headword and/or pronunciation info may be indicated by this attribute.

Attribute value examples: "Kanji_representation", "pronunciation", "original_spelling".

delimiter

(optional) Specifies the symbol to be used when part of the content needs to be placed in the the preceding element for printing.

B.2.3.3 key

Specifies a keyword to search for a dic-item element. Usually used below the headword element that is to be displayed/printed.

NOTE The appearance of a key element is not limited within the head block.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT key (#PCDATA)>
<!ATTLIST key type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Usually, keys without the type attribute are treated as main headword keys. Variant representations and readings of the main headword are specified using type attributes.

Attribute value examples: "representation", "kana", "original_spelling".

An example is shown in Figure B.6

(paper-form)

~ **permíssion [consènt]** *chiefly BrE* official permission to construct a new building or alter an existing one (*AmE* • *AusE* building permit)

```
<dic-item id="P0285900#0000000" rank="compound">
<head>
<headword><spellout
org="&swangacute;">pl&aacute;nning</spellout> perm&igrave;ssion
 [cons&egrave;nt] </headword>
<key type="compound">planning permission</key>
<key type="compound">planning consent</key>
</head>
<meaning><alabel>chiefly BrE</alabel>official permission to
construct a new building or alter an existing one (<alabel>AmE •
AusE</alabel>building permit) </meaning>
</dic-item>
```

Figure B.6 – Examples of the key element

B.2.3.4 snippet

Contains a snippet (excerpt or summary) of the entry which is to be shown onmouse. As logically expected, there shall be at most one instance of snippet element as a child element of the dic-item elment and at most one such instance as a child element of the subhead element. This element is for providing an excerpt view in addition to the regular reading view.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT snippet (#PCDATA)>
<!ATTLIST snippet pid ID #REQUIRED
type CDATA #IMPLIED
headword CDATA #REQUIRED >
```

An example of the markup using this element is shown in Figure B.7. Examples of the regular reading view and snippet view are shown in Figure B.8 and Figure B.9, respectively.

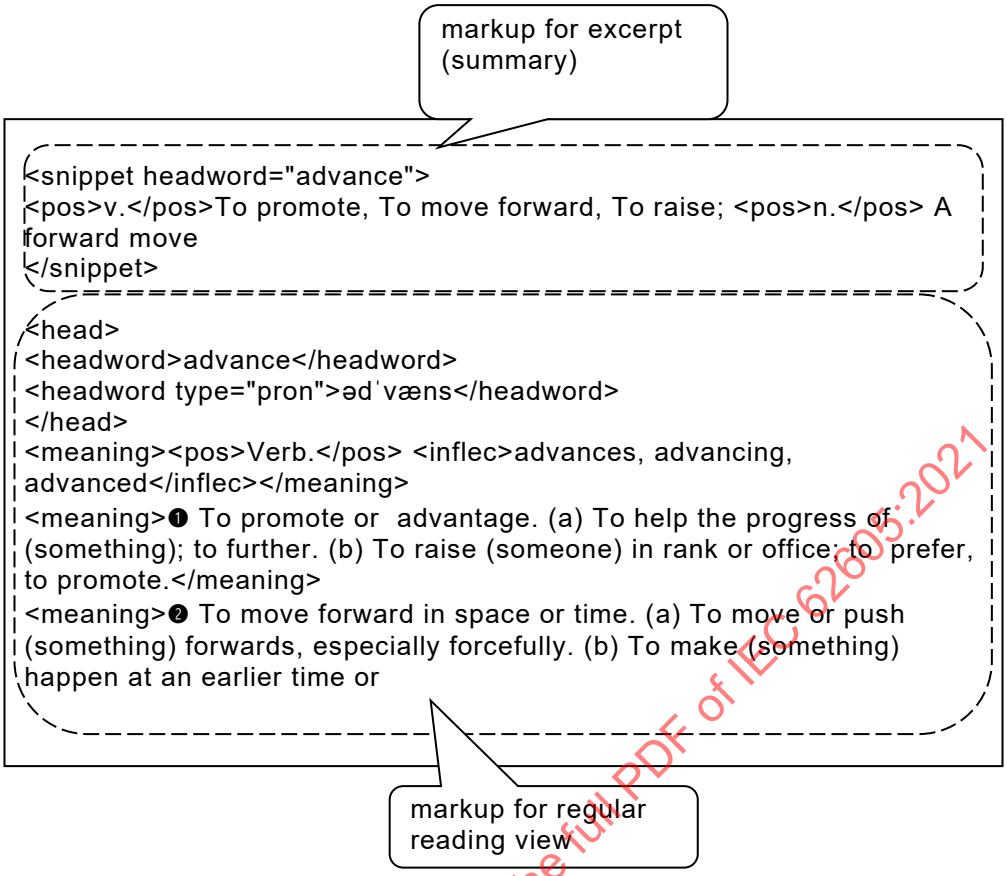


Figure B.7 – Example of the snippet element

advance /ədˈvæns/ Verb. (advances, advancing, advanced)

❶ **To promote** or advantage. (a) To help the progress of (something); to further. (b) To raise (someone) in rank or office; to prefer, to promote. ❷ **To move forward** in space or time. (a) To move or push (something) forwards, especially forcefully. (b) To make (something) happen at an earlier time or date; to bring forward, to hasten. (c) (intransitive) To move forwards; to ap- : move forward in time; to progress towards completion. ❸ **To raise**, be raised. (a) (transitive, now archaic) To raise; to lift or elevate. (b) To raise or increase (a price, rate). (c) To increase :

— Noun (advances) ❶ **A forward move**; improvement or progression: an advance in health or knowledge; an advance in rank or office ❷ An amount of money or credit, especially given as a loan, or paid before it is due; an advancement. :

— Adjective. (more advance, most advance) ❶ completed before need or a milestone event: He made an advance payment on the prior shipment to show good faith. ❷ preceding: The advance

Figure B.8 – Regular reading view

```
<snippet headword="advance">
<pos>v.</pos>To promote, To move forward, To raise; <pos>n.</pos> A
forward move
</snippet>
```

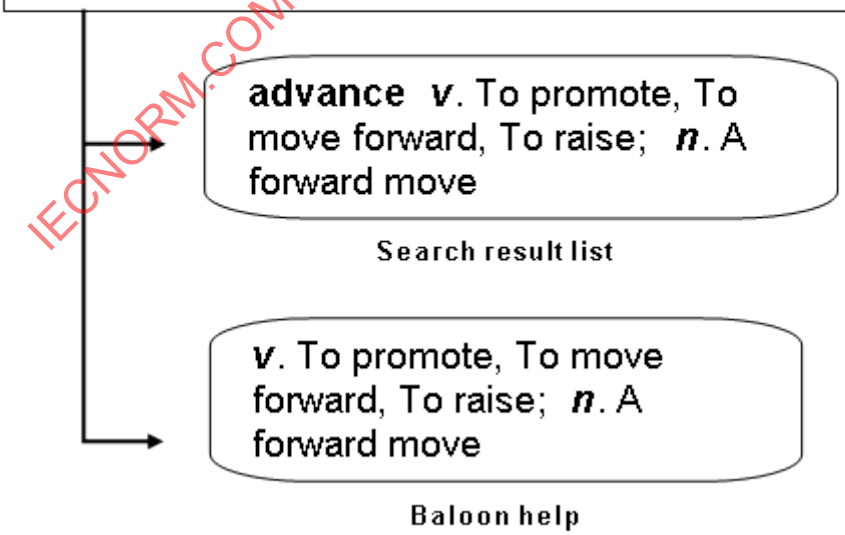


Figure B.9 – Snippet view

B.2.4 Main-text related elements

B.2.4.1 meaning

Specifies word definitions and similar explanations such as usage, notes. Can appear more than once, generally together with example and other elements, which will be explained below.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT meaning (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST meaning subid ID    #IMPLIED
                  type  CDATA #IMPLIED
                  level CDATA #IMPLIED
                  no    CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring.)

type

(optional) Specifies a type of definition.

Attribute value examples: "explanation", "synonym", "etymology", "note".

level

(optional) Specifies the hiearchical level for the definition number in an entry.

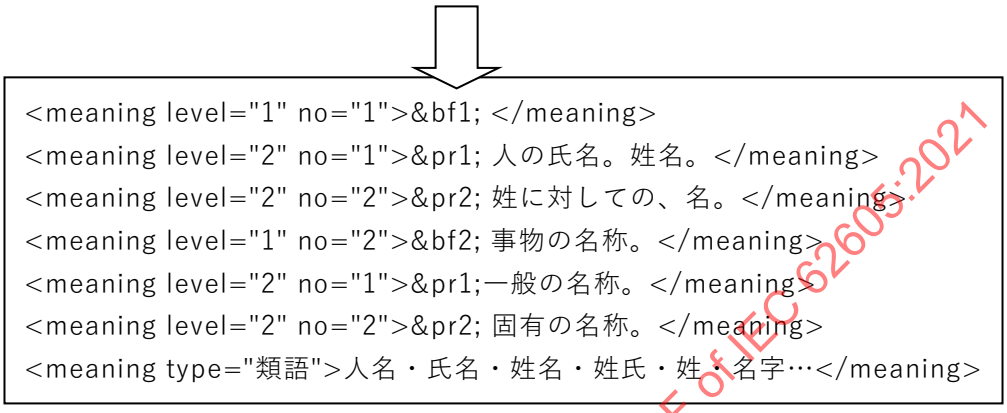
no

(optional) Specifies the definition number.

An example is shown in Figure B.10.

(paper-form)

1 (1) 人の氏名。姓名。(2) 姓に対しての、名。
2 事物の名称。(1) 一般の名称。(2) 固有の名称。
類語 人名・氏名・姓名・姓氏・姓・名字…



```
<meaning level="1" no="1">&bf1; </meaning>
<meaning level="2" no="1">&pr1; 人の氏名。姓名。</meaning>
<meaning level="2" no="2">&pr2; 姓に対しての、名。</meaning>
<meaning level="1" no="2">&bf2; 事物の名称。</meaning>
<meaning level="2" no="1">&pr1;一般の名称。</meaning>
<meaning level="2" no="2">&pr2; 固有の名称。</meaning>
<meaning type="類語">人名・氏名・姓名・姓氏・姓・名字…</meaning>
```

Figure B.10 – Examples of the meaning element

B.2.4.2 meaning-group

Groups definitions and examples by part of speech or meaning category. (The div element, which is retained, has been also used for this purpose)

The syntax of the element is shown in the DTD below and examples are shown in Figure B.11

```
<!ELEMENT meaning-group (meaning | example | subhead | column | key | div)*>
<!ATTLIST meaning-group subid ID    #REQUIRED
                        type  CDATA #IMPLIED
                        level CDATA #IMPLIED
                        no    CDATA #IMPLIED >
```

[Attributes]

subid

Specifies an id for this child element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type for the definition block.

level

(optional) Specifies the position of the definition block in the hierarchy.

no

(optional) Specifies the order for the definition in an ordinal number.

```
<head>
<headword>considering</headword>
</head>
<meaning-group subid="123456#01" type=“preposition">
<meaning><pos>prep.</pos>taking … into consideration, for</meaning>
<examplemarkup for regular reading view
wmarkup for excerpt (summary)
/meaning-group>
<meaning-group subid="123456#02" type=“adverb">
<meaning><pos>adv.</pos>taking everything into
consideration</meaning>
<example> She has played very well, <i>considering</i>. </example>
</meaning-group>
```

Figure B.11 – Examples of the meaning-group element

B.2.4.3 example

Gives examples. Each example should be enclosed in one example element. It shall be repeatable.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT example (#PCDATA | ex-body | ex-trans | ex-src | ex-misc | %inline.html;
| %inline.lexml;)*>
<!ATTLIST example subid     ID    #IMPLIED
                  type     CDATA #IMPLIED
                  delimiter CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Used to distinguish the type of the example when necessary.

delimiter

(optional) Specifies the symbol to be used when part of the content needs to be placed in the preceding element for printing.

An example is shown in Figure B.12.

(paper-form)

mèd-icátion /-kéɪʃən/ [noun] [U] [C] medicine; a drug ‖ The doctor prescribed ~ for the pain. / Her ~s did not agree with her.

```
<dic-item id="M0139700#0000000" rank="03">
<head>
<headword>m&egrave;d-ic&aacute;tion</headword>
<key type="03">medication</key>
<headword type="pronunciation">-k&eacute;
<sc>I</sc>&esh;<i>&schwa;</i>n</headword>
</head>
<meaning
subid="M0139700#NN00000"><pos>noun</pos><pos>U</pos>
<pos>C</pos>medicine; a drug</meaning>
<div type="example">
<example delimiter=" ‖ ">The doctor prescribed <spellout org="~">
medication</spellout> for the pain.</example>
<key
type="example">The;doctor;prescribe;medication;for;the;pain</key>
<example delimiter="/">Her <spellout org="~
s">medications</spellout> did not agree with her.</example>
<key type="example">Her;medication;did;do;not;agree;with;her</key>
```

Figure B.12 – Examples of the example element

Another example is shown in Figure B.13.

(paper-form)

1 遊び道具。おもちゃ。「をかしき絵、―ども」〈源・若紫〉
2 楽器。「多くの―の音」〈源・常夏〉

```
<meaning level="1" no="1">&bf1; 遊び道具。おもちゃ。</meaning>
        <example>「をかしき絵、―ども」〈源・若紫〉</example>
<meaning level="1" no="2">&bf2; 楽器。</meaning>
        <example>「多くの―の音」〈源・常夏〉</example>
```

Figure B.13 – Examples of the meaning element

Yet another example is shown in Figure B.14.

(paper-form)

1 よい，満足できる，すぐれた，りっぱな。‖ a ～ dictionary よい辞書 / ～ land 肥沃な土地
2 正当な；(…に) ふさわしい；(…するのに) 適した《to do》。‖ ～ *for* nothing 《略式》何の役にも立たない / a ～ place *to* live (in) 住みよい場所

```
<meaning level="1" no="1">&bf1; よい，満足できる，すぐれた，りっぱな。</meaning>
        <example delimiter="〃">a ～ dictionary｜よい dictionary</example>
        <example delimiter="/">～ land｜肥沃な土地</example>
<meaning level="1" no="2">&bf2; 正当な；(…に)ふさわしい；(…するのに)適した《to do》。</meaning>
        <example delimiter="〃">～ <i>for</i> nothing｜《略式》何の役にも立たない</example>
        <example delimiter="/">a ～ place <i>to</i> live (in)｜住みよい場所</example>
```

Figure B.14 – Examples of the meaning element

B.2.4.4 example-group

Specifies groups consisting of examples (the div element, which is retained, has been also used for this purpose).

The syntax of the element is shown in the DTD below.

```
<!ELEMENT example-group (example | key)*>
<!ATTLIST example-group subid     ID    #IMPLIED
                        type      CDATA #IMPLIED
                        delimiter CDATA #IMPLIED >
```

[Attributes]

subid

Specifies an id for this child element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Used to distinguish the type of the example group.

delimiter

(optional) Specifies the symbol to be used when part of the content needs to be placed in the preceding element for printing.

Examples are shown in Figure B.15.

```
<headword>tailler</headword>
          :
<meaning>to cut</meaning>
<example-group>
<example>～ un bloc de marbre | to cut a block of marble</example>
<example>～ un crayon | to sharpen a pencil</example>
<example>～ une haie | to prune a hedge</example>
<example>～ étoffe | to cut cloth</example>
</example-group>
<example-group delimiter="◆">
<example>～ A en B | to cut A to make B</example>
<example>～ un diamant en brillant | to cut a diamond into a
brilliant</example>
</example-group>
</meaning>
```

Figure B.15 – Examples of the example-group element

B.2.5 Subheadword related elements

B.2.5.1 subhead

Specifies the subheadword such as a derived word, compound, and related info consisting of meaning, example and key elements in a set.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT subhead (subheadword | key | meaning | example | column | div)*>
<!ATTLIST subhead subid     ID    #IMPLIED
                  type      CDATA #IMPLIED
                  delimiter CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this child element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type of the subheadword.

delimiter

(optional) Specifies the symbol to be used when part of the content needs to be placed in the preceding element for printing.

B.2.5.2 subheadword

Specifies a subheadword.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT subheadword (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST subheadword type      CDATA #IMPLIED
                      delimiter CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type for the subheadword such as a derived word and a compound.

delimiter

(optional) Specifies the symbol to be used when part of the content needs to be placed in the preceding element for printing.

Examples are shown in Figure B.16.

(paper-form)

old /óuld, *AmE+* óuwəld/ orininal meaning: well-nourished, fully grown
— [adjective] (~·er[est]; eld·er /éldər/, eld·est /éldəst/)
1 [usu. after a mumeral] of a specified age ‖ a six-year-~ child (=a child six years ~) / How ~ is she? (=What age is she?)
2 having lived for a lomg time; no longer young (⇔ young); [the ~; pl.] old people
:
5 [attrib.] **a** conventional; long-estabulished; familiar **b** former; belonging to the past
:
àny óld ... ((INFORMAL)) **(1)** any; any item of a particular type ‖ eat *any* ~ thing / Don't give me *any* ~ excuse. **(2)** [not any ~] special or famous; neither common nor ordinary
:
the Old Mán of the Séa
— [noun] [C] [in combination] a human or animal (especially, a bird) of the age specified
:
of óld ((LITERARY)) **(1)** belonging to the past **(2)** in the past; since past times
:

```
<dic-item id="O0050400#0000000" rank="01">
<head>
<headword>old</headword>
        <key>old</key>
<headword type="pronunciation">&oacute;&Usc;ld,
<alabel>AmE+</alabel>&oacute;&Usc;<i>w</i>&schwa;ld</headword>
</head>
<meaning type="etymology">orininal meaning：well-nourished, fully grown</meaning>
<meaning subid="O0050400#AJ00000">—<pos>adjective</pos><inflec>&swang;・
er[est]; eld・er<pron>&eacute;ld&schwa;<i>r</i></pron>, eld・
est<pron>&eacute;ld&schwa;st</pron></inflec></meaning>
<meaning subid="O0050400#AJ01000" level="1" no="01"><b>1</b> [usu. after a
mumeral] of a specified age</meaning>
        <example delimiter="‖">a six-year-<spellout org="&swang;">old</spellout>
child(=a child six years <spellout org="&swang;">old</spellout>)</example>
                <key type="example">a;six-year-old;child</key>
        <example delimiter="/">How <spellout
org="&swangacute;">&oacute;ld</spellout> is she?(=What age is she?)</example>
                <key type="example">How;old;is;be;she</key>
<meaning subid="O0050400#AJ02000" level="1" no="02"><b>2</b> having lived for a
lomg time； no longer young(⇔young); [the &swang;; pl.] old people</meaning>
                                        :
```


Figure B.16 – Examples of the subheadword element *(1 of 2)*

```
                                                    :
<meaning subid="O0050400#AJ05000" level="1" no="05"><b>5</b> [attrib.]</meaning>
<meaning subid="O0050400#AJ05010" level="2" no="a"><b>a</b> conventional; long-
estabulished; familiar</meaning>
                                                    :
<meaning subid="O0050400#AJ05020" level="2" no="b"><b>b</b> former; belonging to
the past</meaning>
                                                    :
<subhead type="Phrase" subid="O0050400#SK00010">
<subheadword type="Phrase">&agrave;ny &oacute;ld ...
<glabel>INFORMAL</glabel></subheadword>
        <key type="phrase">any;old</key>
<meaning level="3" no="01"><b>(1)</b> any; any item of a particular type</meaning>
        <example delimiter="‖">eat <i>any</i> <spellout org="&swang;">old</spellout>
thing</example>
                <key type="example">eat;any;old;thing</key>
        <example delimiter="/">Don't give me <i>any</i> <spellout
org="&swang;">old</spellout> excuse.</example>
                        <key type="example">Don't;do;not;give;me;any;old;excuse</key>

<meaning level="3" no="02"><b>(2)</b> [not any ~] special or famous; neither common
nor ordinary</meaning>
</subhead>
<subhead type="Phrase" subid="O0050400#SK00020">
<subheadword type="Phrase">the Old M&aacute;n of the S&eacute;a</subheadword>
        <key type="phrase">the;Old;Man;of;the;Sea</key>
</subhead>
<meaning subid="O0050400#NN00000">—<pos>noun</pos><pos>C</pos> [in
combination] a human or animal (especially, a bird) of the age specified</meaning>
                                                    :
<subhead type="Phrase" subid="O0050400#SK00030">
<subheadword type="Phrase">of &oacute;ld
<glabel>LITERARY</glabel></subheadword>
```


Figure B.16 *(2 of 2)*

Other examples are shown in Figure B.17.

(paper-form)

condìtionálity noun U [FINANCIAL] conditions for credit
conditionally adverb with conditions attached

```
<subhead type="derivative" subid="ABCD12345600">
<subheadword
type="derivative">cond&igrave;tion&aacute;lity</subheadword>
        <key type="derivative">conditionality</key>
<meaning><pos>noun</pos>  <pos>U</pos>  <slabel>FINANCIAL</slabel>
conditions for credit</meaning>
</subhead>
<subhead type="derivative" subid="ABCD12345700">
<subheadword type="derivative">conditionally</subheadword>
        <key type="derivative">conditionally</key>
<meaning><pos>adverb</pos> with conditions attached</meaning>
</subhead>
```

Figure B.17 – Examples of the subheadword elements

Yet other examples are shown in Figure B.18.

(paper-form)

con·di·tion·al·i·ty 〔名〕〔U〕条件［制限］つき。
con·di·tion·al·ly 〔副〕条件つきで。

```
<subhead type="派生" subid="ABCD12345600">
<subheadword type="派生">con・d&igrave;・tion・&aacute;l・i・
ty</subheadword>
        <key type="派生">conditionality</key>
<meaning><pos>名</pos><pos>U</pos>条件［制限］つき。
</meaning>
</subhead>
<subhead type="派生" subid="ABCD12345700">
<subheadword type="派生">con・di・tion・al・ly</subheadword>
        <key type="派生">conditionally</key>
<meaning><pos>副</pos>条件つきで。</meaning>
```

Figure B.18 – Examples of the subheadword element

B.2.6 Other block elements

B.2.6.1 index

A block element used for expressing various indices. Alternatively, the column element can be used for this purpose with the type attribute being set to "index".

The syntax of the element is shown in the DTD below.

```
<!ELEMENT index (meaning | indexlist)*>
<!ATTLIST index subid ID    #IMPLIED
            type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type for the index.

B.2.6.2 indexlist

Gives the detailed information for the index.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT indexlist (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST indexlist type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type for the index.

Examples are shown in Figure B.19.

(paper-form)

institution

original meaning：someting which has been established
・establishment (noun 3)
├ structure and organization (noun 1) ……an organization founded for a particular purpose
└ custom (noun 2) ……an established practice

—— noun

1 C a large and important organization, such as a university or bank; the building which the organization possesses ‖ an ***educational*** [academic] ***institution*** / *institutions* of higher education / a mental *institution*

2 C a custom or system that has existed for a long time ‖ political *institutions* / the *institution* of marriage

3 U founding, foundation, setting up, establishment, start ‖ the *institution* of a bank

：

```
<dic-item id="I0182700#00000" rank="02">
<head>
<headword>institution</headword>
<key>institution</key>
</head>
<index>
<meaning type="etymology">original meaning: someting which has been
established</meaning>
<indexlist type="level1">establishment (<ref
refid="I0182700#NN030"><pos>noun</pos> 3</ref>)</indexlist>
<indexlist type="level2">structure and organization (<ref
refid="I0182700#NN010"><pos>noun</pos> 1</ref>）……an organization founded
for a particular purpose</indexlist>
<indexlist type="level2">custom (<ref refid="I0182700#NN020"><pos>noun</pos>
2</ref>）……an established practice</indexlist>
</index>
<meaning subid="I0182700#NN000">—<pos>noun</pos></meaning>
<meaning subid="I0182700#NN010"><b>1</b> <pos>C</pos> a large and
important organization, such as a university or bank; the building which the
organization possesses</meaning>
<example>an <em>educational</em> [academic] <em>institution</em></example>
<example><i>institutions</i> of higher education</example>
<example>a mental <i>institution</i></example>
<meaning subid="I0182700#NN020"><b>2</b> <pos>C</pos> a custom or system
that has existed for a long time</meaning>
<example>political <i>institutions</i></example>
<example>the <i>institution</i> of marriage</example>
<meaning subid="I0182700#NN030"><b>3</b> <pos>U</pos> founding,
foundation, setting up, establishment, start</meaning>
<example>the <i>institution</i> of a bank</example>
. . . . . .
</dic-item>
```


Figure B.19 – Examples of the indexlist element

B.2.6.3 column

Denotes the text data that comprises a boxed article or column.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT column (title | key | meaning | example | subhead)*>
<!ATTLIST column subid ID    #IMPLIED
```

```
type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type.

Attribute value examples: "usage", "synonyms", "culture", "rel-inf" (abbreviation for "related information").

B.2.6.4 div

Specifies general-purpose block elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT div (title | key | meaning | example | subhead)*>
<!ATTLIST div subid ID    #IMPLIED
              level CDATA #IMPLIED
              type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this child element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type for the block.

NOTE An example of the type attribute introducing mathematical formulae is given in B.3.10.

level

(optional) Specifies a position in the hierarchy.

B.2.6.5 title

Specifies title for the block defined by column or div elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT title (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST title type  CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type for the title.

B.2.7 Media related elements

B.2.7.1 image

Specifies a non-inline image file (for inline image files, use an img element instead.)

The syntax of the element is shown in the DTD below.

```
<!ELEMENT image EMPTY>
<!ATTLIST image type      CDATA #IMPLIED
                src       CDATA #REQUIRED
                mime-type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type for the image.

src

Specifies the file name.

mime-type

(optional) specifies the MIME type.
Attribute value examples: "image/gif", "image/jpeg", "image/png".

B.2.7.2 audio

Specifies an audio source file.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT audio EMPTY>
<!ATTLIST audio type CDATA #IMPLIED
        src      CDATA #REQUIRED
        mime-type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type of the audio (such as audio content).

src

Specifies the file name.

mime-type

(optional) Specifies the MIME type.
Attribute value examples: "audio/mpeg", "audio/vnd.rn-realaudio", "audio/wav".

B.2.7.3 video

Specifies a movie file. Can be used to introduce either inline or non-inline video objects.

[Child elements]

The syntax of the element is shown in the DTD below.

```
<!ELEMENT video EMPTY>
<!ATTLIST video type     CDATA #IMPLIED
        src      CDATA #REQUIRED
        mime-type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type of the video (such as video content).

src

Specifies the file name.

mime-type

(optional) Specifies the MIME type.

Attribute value examples: "video/mpeg", "video/x-msvideo", "application/x-shockwave-flash".

B.2.8 Other structural elements

B.2.8.1 p

Specifies paragraphs.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT p (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST p subid ID    #IMPLIED
        type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this child element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type of the text data.

B.2.8.2 ul, li

B.2.8.2.1 ul

Used to introduce an unordered list.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ul (li)+>
<!ATTLIST ul subid ID    #IMPLIED
        type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type for the list.

B.2.8.2.2 li

Used to introduce an item in the list.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT li (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST li subid ID    #IMPLIED
             type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type for the item.

B.2.8.3 dl, dt, dd

B.2.8.3.1 dl

Definition list.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT dl (dt|dd)+>
<!ATTLIST dl subid ID    #IMPLIED
             type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type.

B.2.8.3.2 dt

The syntax of the element is shown in the DTD below.

```
<!ELEMENT dt (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST dt subid ID    #IMPLIED
        type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type.

B.2.8.3.3 dd

The syntax of the element is shown in the DTD below.

```
<!ELEMENT dd (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST dd subid ID    #IMPLIED
        type  CDATA #IMPLIED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

type

(optional) Specifies the type.

B.2.8.4 table, caption, th, tr, td

B.2.8.4.1 table

Specifies a table.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT table (caption? | tr+)>
<!ATTLIST table
    %subid.attr;
    %type.attr;
>
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring).

B.2.8.4.2 caption

The syntax of the element is shown in the DTD below.

```
<!ELEMENT caption (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST caption type  CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type.

B.2.8.4.3 tr

The syntax of the element is shown in the DTD below.

```
<!ELEMENT tr (th | td)*>
```

[Attributes]

None.

B.2.8.4.4 th

The syntax of the element is shown in the DTD below.

```
<!ELEMENT th (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST th colspan NMTOKEN #IMPLIED >
```

[Attributes]

colspan

(optional) Specifies the number of columns spanned by the current cell.

B.2.8.4.5 td

The syntax of the element is shown in the DTD below.

```
<!ELEMENT td (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST td colspan NMTOKEN #IMPLIED
        rowspan NMTOKEN #IMPLIED >
```

[Attributes]

colspan

(optional) Specifies the number of columns spanned by the current cell.

rowspan

(optional) Specifies the number of rows spanned by the current cell.

B.2.8.5 replace

Specifies the file that can replace the text enclosed by the src attribute.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT replace (meaning | example | subhead)*>
<!ATTLIST replace subid ID    #IMPLIED
            type  CDATA #IMPLIED
            src   CDATA #REQUIRED >
```

[Attributes]

subid

(optional) Specifies an id for this element, usually derived from the id attribute value of the ancestor dic-item element (typically the former including the latter as a substring.

type

(optional) Specifies the type.

src

As a principle, image file name that can be replaced with the text in the replace block is to be specified. The file name extension is needed so that the file can be handled without MIME type.

An example is shown in Figure B.20.

```
<replace type="語義の展開" src="admit.jpg">
<meaning>「運ぶ」&bc7;┬「身につける」&bc2;─「心に持つ」
&bc4;</meaning>
<meaning>              ├(重さを)「支える」&bc3;─「耐える」
&bc1;</meaning>
<meaning>              └(もたらす)─「産む」&bc5;</meaning>
</replace>
```

Figure B.20 – Example of the replace element

B.2.8.6 memo

Specifies memo data for editing and management purposes.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT memo (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST memo type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type.

B.2.8.7 data

Specifies infomation for database processing such as categorization and sorting information.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT data (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST data type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type.

B.3 Inline elements

B.3.1 Labels

B.3.1.1 alabel

Denotes regional label.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT alabel (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST alabel type CDATA #IMPLIED
                 code CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies a type.

code

(optional) Specifies the coding/categorization system and the code value on it.

The following are examples of such coding/categorization systems.

"ISO3166" ISO country codes
"ccTLD" country code top-level domain
"IOC" IOC (International Olympic Committee) country codes
"JISX0402" Japan's local government codes

An example is shown in Figure B.21.

```
<alabel>usu. AmE</alabel>
```

Figure B.21 – Example of the alabel element

Other examples are shown in Figure B.22.

```
<headword>agenciero</headword>
        :
<meaning>①<alabel code="ISO3166-CUB">CUBA</alabel>・<alabel
code="ISO3166-MEX">MEXICO</alabel>moving firm</meaning>
<meaning>②<alabel code="ISO3166-
ARG">Argentine</alabel>agent</meaning>
<meaning>③<alabel code="ISO3166-CHL">CHILE</alabel>Pawnshop
owner</meaning>
```

Figure B.22 – Examples of the alabel element

B.3.1.2 glabel

Denotes grammatical/speech label.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT glabel (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST glabel type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies a type.

Attribute value examples: "word-form", "attributive".

An example is shown in Figure B.23.

```
<glabel>INFORMAL</glabel>
```

Figure B.23 – Example of the glabel element

Another example is shown in Figure B.24.

```
<headword>your</headword>
        :
<meaning><pos>pron</pos> <glabel type="限定">you の所有格</glabel> <b>あなた
(がた) の</b>; <glabel type="語層">話</glabel> 例の, いわゆる; <glabel type="語形">Y-</glabel> 貴人への呼びかけ.</meaning>
```

Figure B.24 – Examples of the glabel element

B.3.1.3 slabel

Denotes category of the terms.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT slabel (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST slabel type  CDATA #IMPLIED
                genre CDATA #IMPLIED
                code  CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies a type.

genre

(optional) Specifies the categorization system and the category on it.

code

(optional) Specifies the coding/categorization system and the code value on it.

An example is shown in Figure B.25.

```
<slabel>MEDICAL</slabel>
```

Figure B.25 – Example of the slabel element

Yet other examples are shown in Figure B.26.

```
<slabel code="LCC:QL461-599.82">Insects</slabel>
<slabel genre="LCC:SCIENCE_Zoology_Invertebrate">Insects</slabel>
```

Figure B.26 – Examples of the slabel element

B.3.2 pronunciation/accent related elements

B.3.2.1 accent

Accent information. To be used for description of accent and intonation in other places than headwords and pronunciation information.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT accent (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST accent type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type.

An example is shown in Figure B.27.

```
<headword type="表記">愛育</headword>
        :
<meaning><pos>名・する</pos><accent>ア<b>イイク</b></accent>子どもなどをかわいがって
育てること。</meaning>
```

Figure B.27 – Example of the accent element

B.3.2.2 pron

Specifies pronunciation information in inline manner. Pronunciation information of the headword should be basically written in headword elements with type attribute specified accordingly.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT pron (#PCDATA)>
```

[Attributes]

None.

B.3.2.3 pha

Provides pronunciation information based on the phonetic alphabet (thus the name "pha"). While the type attribute can specify the type such as IPA and pinyin, dedicated elements can alternatively be used.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT pha (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST pha type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type as described above.

Attribute value examples: "ipa", "pinyin", "respelling", "us_respelling".

B.3.2.4 ipa

Provides pronunciation information based on the International Phonetic Alphabet.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ipa (#PCDATA | %inline.html; | %inline.lexml;)*>
```

[Attributes]

None.

Examples are shown in Figure B.28

```
<headword type="見出">ABC<sup>1</sup></headword>
<headword type="発音
"><ipa>&eacute;ib&igrave;&length;s&iacute;&length;</ipa>
<b>エ</b>イビー<b>スィ</b>ー</headword>
        :
<meaning><POS>名</POS> （複 ABC's, ABCs <pron><ipa>-z</ipa></pron>）
</meaning>
```

Figure B.28 – Examples of the ipa element

B.3.2.5 pinyin

Provides pronunciation information based on pinyin (the Chinese phonetic alphabet), mainly used in Chinese language contents.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT pinyin (#PCDATA | %inline.html; | %inline.lexml;)*>
```

[Attributes]

None.

B.3.3 Part-of-speech related and other elements

B.3.3.1 pos

Denotes a part of speech. Other symbols can be specified here.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT pos (#PCDATA)>
```

[Attributes]

None.

B.3.3.2 gender

Gramatical gender info. Alternatively the pos element can include such information instead of using this element, depending on the policy of the publisher.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT gender (#PCDATA)>
```

[Attributes]

None.

B.3.4 Example-related elements

B.3.4.1 ex

Denotes inline usage/phrase examples.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ex (#PCDATA | ex-body | ex-trans | ex-src | ex-misc | %inline.html; | %inline.lexml;)*>
<!ATTLIST ex type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type.

B.3.4.2 ex-body

Denotes the body of the example. To be used in the example and ex elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ex-body (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST ex-body lang CDATA #IMPLIED
                 type CDATA #IMPLIED >
```

[Attributes]

lang

(optional) Specifies a language code.

type

(optional) Specifies the type.

B.3.4.3 ex-trans

Denotes a translation of the example (the ex-body element). To be used in the example and ex elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ex-trans (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST ex-trans lang CDATA #IMPLIED
                   type CDATA #IMPLIED >
```

[Attributes]

lang

(optional) Specifies a language code.

type

(optional) Specifies the type.

B.3.4.4 ex-src

Denotes the source of the example (the ex element). To be used in the example and ex elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ex-src (#PCDATA | ex-src-author | ex-src-title | %inline.html; | %inline.lexml;)*>
<!ATTLIST ex-src refid IDREF #IMPLIED
                 type  CDATA #IMPLIED >
```

[Attributes]

refid

(optional) Specifies the id attribute value of the source.

type

(optional) Specifies the type.

B.3.4.5 ex-misc

Denotes miscellaneous information concerning the example (the ex element). To be used in the example and ex elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ex-misc (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST ex-misc type CDATA #IMPLIED >
```

[Attributes]

type

(optional) Specifies the type.

B.3.4.6 ex-src-author

Denotes author information concerning the example (the ex element). To be used in the example and ex elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ex-src-author (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST ex-src-author org CDATA #IMPLIED >
```

[Attributes]

org

(optional) Specifies the long form of the abbreviated author name of the source.

B.3.4.7 ex-src-title

Denotes title information concerning the example (the ex element). To be used in the example and ex elements.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT ex-src-title (#PCDATA | %inline.html; | %inline.lexml;)*>
<!ATTLIST ex-src-title org CDATA #IMPLIED >
```

[Attributes]

org

(optional) Specifies the long form of the abbreviated title of the source.

Examples of the ex-body, ex-trans, ex-src, ex-src-author and ex-src-title elements are shown in Figure B.29.

```
<example>
<ex-body lang="en">There is a tide in the affairs of men.</ex-body>
<ex-trans lang="zh">人生事业有盛有衰</ex-trans>
<ex-src><ex-src-author>Shak.</ex-src-author><ex-src-title>Caes.</ex-
src-title> IV.iii.217</ex-src>
</example>
```

Figure B.29 – Examples of the ex- elements

B.3.5 Other dictionary-specific elements

B.3.5.1 abbr

When the headword is not in an abbreviated form, the corresponding abbreviation is given by the abbr element.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT abbr (#PCDATA | %inline.html; | %inline.lexml;)*>
```

[Attributes]

None.

An example is shown in Figure B.30.

```
<headword>Lexicographical Extensible Markup Language</headword>
        :
<meaning><abbr>LeXML</abbr>. 辞書・事典に特化した XML 仕様。
</meaning>
```

Figure B.30 – Examples of the abbr element

B.3.5.2 article

To specify the definite article(s) that is/are written in the headword but to be ignored for sorting purposes.

The syntax of the element is shown in the DTD below.

```
<!ELEMENT article (#PCDATA | %inline.html; | %inline.lexml;)*>
```

[Attributes]

None.