

UOML (Unstructured Operation Markup Language) Part 1 1.0 revised by Errata CD02

OASIS Working Draft 02

9 October 2010

Specification URIs: This Version:

Previous Version:

<http://docs.oasis-open.org/uoml-x/v1.0/os/uoml-part1-v1.0-os.html>

<http://docs.oasis-open.org/uoml-x/v1.0/os/uoml-part1-v1.0-os.odt>

<http://docs.oasis-open.org/uoml-x/v1.0/os/uoml-part1-v1.0-os.pdf> (Authoritative)

Latest Version:

<http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-part1-v1.0-errata.html>

<http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-part1-v1.0-errata.pdf> (Authoritative)

Technical Committee:

OASIS Unstructured Operation Markup Language Extended (UOML-X) TC

Chair(s):

Alex Wang, Sursen Corporation <alexwang@sursen.com>

Allison Shi, Sursen Corporation <allison_shi@sursen.com> (since September 2007)

Editor(s):

Joel Marcey, Sursen Corporation <joel@sursen.com>

Ningsheng Liu, Sursen Corporation <lns@sursen.com>

Kaihong Zou, Sursen Corporation <zoukaihong@sursen.com>

Previous Editor(s):

Xu Guo, Sursen Corporation <guoxu@sursen.com>

Allison Shi, Sursen Corporation <allison_shi@sursen.com>

Pine Zhang, UOML Alliance <pine_zhang@sursen.com>

Related work:

[N/A]

Normative UOML XML Schema Location:

<http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-part1-v1.0-schema-errata.xsd>

Declared XML Namespace(s):

urn:oasis:names:tc:uoml:xmlns:uoml-x:1.0

Abstract:

This specification defines the Unstructured document Operation Markup Language (UOML), a platform-neutral operation interface that allows applications to dynamically access and update the visual appearance of fixed layout documents.

UOML provides a standard set of objects for representing fixed layout documents (or the fixed layout of documents), describes how these objects can be organized, and defines a standard set of operations for accessing and manipulating them.

Document service vendors can support UOML as an interface to their proprietary documents; content authors can write to the standard UOML interfaces rather than vendor-specific APIs, thus increasing the interoperability of document software.

Status:

This document was last revised or approved by the OASIS Unstructured Operation Markup Language eXtended (UOML-X) Technical Committee on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/uoml-x/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<http://www.oasis-open.org/committees/uoml-x/ipr.php>).

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/uoml-x/>.

Notices

Copyright © OASIS® 2010. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

Table of Contents

1. Introduction	6
1.1 Terminology.....	7
1.2 Scope	10
1.3 Notational Conventions	11
1.4 Acronyms and Abbreviations	12
1.5 General Description	13
1.6 Overview	14
1.7 Normative References	15
1.8 Non-Normative References.....	17
2. Abstract Document Model	18
2.1 Overview	18
2.2 Docbase.....	19
2.3 Docset	19
2.4 Document	19
2.5 Font	20
2.6 Page.....	20
2.7 Layer.....	20
2.8 Object Stream.....	21
2.9 Graphics Object	21
2.10 Command Object.....	21
2.11 UML Diagram of UOML.....	21
2.12 Page Rendering Model.....	22
3. UOML Instructions.....	23
3.1 OPEN	23
3.2 CLOSE	24
3.3 USE	24
3.4 GET	25
3.5 SET	28
3.6 INSERT.....	28
3.7 DELETE.....	29
3.8 SYSTEM.....	30
3.9 RET.....	30
4. UOML Objects	32
4.1 Logical Coordinate System and Units	32
4.2 Graphics State.....	32
4.3 DOCBASE	32
4.4 DOCSET.....	33
4.5 DOC.....	33
4.5.1 Metadata	33
4.6 FONT DEFINITION.....	33
4.6.1 FONTLIST	34
4.6.2 FONTMAP.....	34
4.6.3 EMBEDFONT.....	34
4.7 PAGE.....	34

4.8	LAYER.....	34
4.9	OBJSTREAM.....	34
4.10	Graphics Objects.....	35
4.10.1	ARC.....	35
4.10.2	BEZIER.....	35
4.10.3	CIRCLE.....	36
4.10.4	ELLIPSE.....	36
4.10.5	IMAGE.....	36
4.10.6	LINE.....	37
4.10.7	RECT.....	37
4.10.8	ROUNDRECT.....	37
4.10.9	SUBPATH.....	37
4.10.10	PATH.....	38
4.10.11	TEXT.....	39
4.10.12	Coordinate and subpath Encoding Rules.....	39
4.11	Command Object.....	41
4.11.1	CMD.....	41
4.11.2	Values for CMD's 'name' property.....	42
4.11.3	Definition of Referenced Type.....	56
4.12	Default Value of Graphics State.....	57
4.13	Definition of Parameter Data Types.....	58
4.13.1	INT.....	58
4.13.2	DOUBLE.....	58
4.13.3	LONG.....	58
4.13.4	DATE.....	58
4.13.5	TIME.....	58
4.13.6	DATETIME.....	59
4.13.7	DURATION.....	59
4.13.8	STRING.....	59
4.13.9	BINARY.....	59
4.13.10	BOOL.....	59
4.13.11	COMPOUND.....	59
4.14	Data Ranges.....	60
5.	Conformance.....	62
5.1.1	DCMS Conformance.....	62
5.1.2	Application Conformance.....	62
Annex A.	UOML XML Schema.....	63
Annex B.	Detailed UOML Examples.....	75
Annex C.	RELAX NG Representation of the UOML XML Schema.....	82
Annex D.	Acknowledgements.....	90

1. Introduction

This text is informative

This OASIS standard specifies an XML schema, called the *Unstructured Operation Markup Language*, which defines an XML-based instruction set to access the visual appearance of unstructured documents and associated information.

This OASIS standard specifies an operation interface for accessing and manipulating the visual appearance of documents. It first defines an abstract document model, which is a set of standard objects and the way they are organized. Secondly, it defines a set of standard operations as an interface to access and manipulate these objects.

In the Unstructured Operation Markup Language (UOML), the term “document” is restricted to its visual appearance. With UOML, programmers can build, modify, and manage documents and their contents. UOML provides a unified interface to access and manipulating documents that simplifies the work to access them.

The goal of UOML is to enable the implementation of the UOML interface by the widest set of tools and platforms; thus fostering interoperability across multiple vendors, applications and platforms. There are two types of UOML implementations: Docbase Management System (DCMS) implementations that execute UOML instructions and application software implementations that issues UOML instructions.

UOML is valuable for document interoperation. Document editing software usually processes documents in its own proprietary format. With UOML, operation on a document is performed through a DCMS Document editing software can cooperate with multiple DCMS and can edit a document regardless of its format. Conversely, a DCMS can cooperate with various document-editing software. Thus, interoperability is achieved.

With the help of UOML, document-editing software can put its focus on editing functionality and need not handle document formats, while a DCMS can put its focus on the functionality and performance of document operation and need not care about specific software applications. Industry division is thus realized, and free market competition is encouraged.

End of informative text

27

28 1.1 Terminology

29 For the purposes of this document, the following terms and definitions apply. Other terms are defined where
30 they appear in *italics* typeface. Terms not explicitly defined in this OASIS standard are not to be presumed to
31 refer implicitly to similar terms defined elsewhere.

32 Throughout this OASIS standard, the terminology “must”, “must not”, “required”, “shall”, “shall not”, “should”,
33 “should not”, “recommended”, “may” and “optional” in this document shall be interpreted as described in
34 RFC 2119, *Keywords for use in RFCs to Indicate Requirement Levels*. [RFC2119].

35

36 **DCMS:** Abbreviated for “Docbase Management System”.

37 **docbase:** The root level of the UOML abstract document model. Abbreviated for “document base”, it is the
38 container of one or many documents. A docbase contains one and only one root docset. [*Note:* The docbase is
39 analogous to a file system on a modern operating system. The term docbase is derived from the term
40 “database”. The docset is analogous to a directory within a file system on a modern operating system. The root
41 docset is analogous to the root directory of a file system. *end note*].

42 **Docbase Management System:** The software that implements the functionality defined by the UOML
43 specification. Abbreviated as DCMS.

44 **docset:** A set of documents. A docset may contain one to many docsets. [*Note:* The docset is analogous to a
45 directory within a file system on a modern operating system. *end note*].

46 **document global object:** A document global object may include a fontlist, fontmap and/or embedfont.

47 **graphics object:** An object that is drawable by the render engine. It describes part or all of the appearance on a
48 page. Examples include images and text.

49 **graphics state:** An internal structure maintained by the DCMS to hold current graphics control parameters. A
50 command object changes one or multiple parameters in the current graphics state.

51 **graphics state stack:** A sequence of graphics states where the first one in is the last one out. A DCMS shall
52 maintain a stack for graphics states, called the graphics state stack. [*Note:* The command object PUSH_GS
53 saves a copy of the current graphics state onto the stack. The command object POP_GS restores the saved
54 copy, remove it from the stack and make it the current graphics state. *end note*]

55 **Implementation-dependent:** indicates an aspect of this specification that may differ between implementations,
56 is not specified by this specification, and is not required to be specified by the implementer for any particular
57 implementation.

58 **layer:** A page is composed of one or more layers. A layer has the same size as the page on which it is
59 constructed. The visual appearance of a page is a combination of all of the layers of the page.

60 **object:** The UOML abstract document model is a tree structure, and a node in the tree is called a UOML object,
61 abbreviated as object.

62 **object stream:** A sequence of graphics objects and command objects. A layer holds object streams.

63 **page bitmap:** A raster image that represents the visual appearance of the page. The number of pixels of the
64 raster image depends on the resolution of the raster image. The number of pixels in the horizontal direction
65 equals the page width multiplied by the resolution; the number of pixels in the vertical direction equals the
66 page height multiplied by the resolution. [*Note:* The resolution is the same for both the horizontal and vertical
67 direction. *end note*]

68 **Path:** A Path is a graphics object composed of straight and/or curved line segments, which may or may not be
69 connected. [*Note:* that in this document, 'path' (all lowercase) refers to a filename, location of docbase or
70 image file. This is different from this current definition of "Path" (with the uppercase 'P'). *end note*]

71 **position number:** Integer starting at 0 to some implementation-dependent maximum, which defines a sequence
72 of objects. [*Note:* the order of a specific sub-object amongst all sub-objects belong to same parent object. It is a
73 continual integer starting at 0 *end note*]
74

75 **sub-element:** In a UOML object XML representation, a sub-element is the child XML node of its parent XML
76 node. [*Note:*

77
78 In UOML a sub-element is a child XML element in the UOML object's XML representation. For example, the
79 XML representation of a CMD object in UOML could be:

80
81 <CMD name="COLOR_LINE" >
82 <rgb r="128" g="3" b="255" a="120"/>
83 </CMD>
84

85 where rgb is a sub element of CMD.

86
87 *end note*]

88 **sub-object:** In the UOML abstract document model tree structure instance, a sub-object is the child node of its
89 parent object node. Each sub-object has only one parent node. A parent node may have multiple sub-objects as
90 child nodes. [*Note:* A sub-object is created by the UOML INSERT instruction. A sub-object describes part of the
91 logical model of the UOML object tree. For example, a logical model of a document could be:

92
93 docbase
94 docset
95 document
96 page
97 layer
98 object stream
99

100 where the child object is the sub-object of the parent object. For example, document is the sub-object of docset,
101 page is the sub-object of document, etc. However, there is no single XML representation of the whole UOML
102 docbase since UOML does not specify the format of document. The XML schema of each UOML object
103 describes the object itself, not including its sub-object, and should only be used as a part of a UOML instruction.
104 *end note*]

105

106

UOML: abbreviation of "Unstructured Operation Markup Language".

107

108 **1.2 Scope**

109 This OASIS standard describes the abstract document model of UOML and the operations available on it.
110 Specifically, operations providing functionality for read/write/edit and display/print on layout-based
111 documents are described. This standard does not define any binding for the operations on the UOML document
112 model. Such bindings are implementation-defined or will be defined in other parts of this standard.

113

114 1.3 Notational Conventions

115 The following typographical conventions are used in this OASIS standard:

- 116 1. The first occurrence of a new term is written in italics, as in "*normative*".
- 117 2. In each definition of a term in §1.1 (Terminology), the term is written in bold, as in "**docset**".

118

119 **1.4 Acronyms and Abbreviations**

120 **This clause is informative**

121 The following acronyms and abbreviations are used throughout this OASIS standard:

122 DCMS — Docbase Management System

123 IEC — the International Electrotechnical Commission

124 ISO — the International Organization for Standardization

125 UOML — Unstructured Operation Markup Language

126 W3C — World Wide Web Consortium

127 **End of informative text**

128

129 1.5 General Description

130 This OASIS standard is divided into the following subdivisions:

- 131 1. Front matter (clause 1);
- 132 2. Main body (clauses 2-4);
- 133 3. Conformance (clause 5);
- 134 4. Annexes

135 Examples are provided to illustrate possible forms of the constructions described. References are used to refer
136 to related clauses. Notes may be provided to give advice or guidance to implementers or programmers.

137 The following form the normative pieces of this OASIS standard:

- 138 • Clauses 1 (except sub-clauses 1.4, 1.6, and 1.8) and 2–5

139 The following form the informative pieces of this OASIS standard:

- 140 • Introductory text in clause 1
- 141 • Sub-clauses 1.4, 1.6, and 1.8
- 142 • All annexes
- 143 • All notes and examples

144 Except for whole clauses or annexes that are identified as being informative, informative text that is contained
145 within normative text is indicated in the following ways:

- 146 1. [*Example*: code fragment, possibly with some narrative ... *end example*]
- 147 2. [*Note*: narrative ... *end note*]
- 148 3. [*Rationale*: narrative ... *end rationale*]
- 149 4. [*Guidance*: narrative ... *end guidance*]

150

151 1.6 Overview

152 This clause is informative

153 This OASIS standard specifies an instruction set of XML elements and attributes describing operations on
154 unstructured, fixed-layout documents. These instructions are for the processing of these documents to
155 accomplish various functionality, such as display and edit.

156 UOML is to unstructured documents as SQL (Structured Query Language) is to structured data. UOML is
157 expressed using standard XML via an instance of an XML schema. UOML handles fixed-layout documents and
158 its associated information (e.g., metadata, security rights, etc.) Fixed-layout documents are two-dimensional
159 and contain static paging information (i.e., information that can be recorded on traditional paper). Thus, the
160 document stores fixed-layout 2D static information that describes the visual appearance.

161

162 Software that implements a conforming implementation of the UOML specification is called a DoCbase
163 Management System (DCMS). Applications process a UOML document by sending UOML instructions
164 (operations) to the DCMS.

165

166 The UOML graphics object model is similar to the graphics model specified by ISO/IEC 32000-1:2008, the
167 Portable Document Format (PDF) standard. For example, both standards describe a page layout using logical
168 coordinate systems, and the positions of the graphics objects are specified using coordinates in the logical
169 coordinate systems. The similarity of the two models allows UOML to be used as an interface standard for PDF.

170 This OASIS standard forms the foundation of UOML. Other standards building upon this standard may be
171 created in the future.

172

173 End of informative text

174

175

176 1.7 Normative References

177 The following referenced documents are indispensable for the interpretation of this document. For dated
178 references, only the edition cited applies. For undated references, the latest edition of the referenced
179 document (including any amendments) applies.

180

181 **[FloatingPoint]** ANSI/IEEE 754-1985, *Standard for Binary Floating-Point Arithmetic*.
182 <http://ieeexplore.ieee.org/servlet/opac?punumber=2355>.

183 **[BMP]** Bitmap Format. BMP. <http://msdn.microsoft.com/en-us/library/at62haz6.aspx>

184 **[RGB]** IEC 61966-2-1: 1999: Multimedia systems and equipment — Colour measurement and management —
185 Part 2-1: Colour management — Default RGB colour space — sRGB. International Electrotechnical
186 Commission, 1999. ISBN 2-8318-4989-6 as amended by Amendment A1:2003. **[DATE]** ISO 8601:2004, *Data*
187 *elements and interchange formats – Information Interchange – Representation of dates and times*.

188 **[DATATYPES]** ISO 11404:2006, *Information Technology – General Purpose Datatypes*.

189 **[TIFF]** ISO 12639:2004, *Graphic technology — Prepress digital data exchange — Tag image file format for*
190 *image technology (TIFF/IT)*.

191 **[Vocabulary]** ISO/IEC 2382-1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*.

192 **[JPEG]** ISO/IEC 10918, *Information technology — Digital Compression and Coding of Continuous-Tone Still*
193 *Images*.

194 **[JBIG]** ISO/IEC 11544, *Information technology — Coded Representation of Picture and Audio Information —*
195 *Progressive Bi-Level Image Compression*.

196 **[IANA-CHARSETS]** (*Internet Assigned Numbers Authority*) *Official Names for Character Sets*, ed. Keld
197 Simonsen et al, <http://www.iana.org/assignments/character-sets>

198 **[OpenFont]** ISO/IEC 14496-22:2007, *Information technology — Coding of Audio-Visual Objects — Part 22:*
199 *Open Font Format*.

200 **[BNF]** ISO/IEC 14977:1966, *Information technology — Syntactic metalanguage — Extended BNF*.

201 **[PNG]** ISO/IEC 15948:2004, *Information technology — Computer Graphics and Image Processing – Portable*
202 *Network Graphics (PNG)*.

203 **[RFC2119]** RFC 2119 *Keywords for use in RFCs to Indicate Requirement Levels*, The Internet Society,
204 Bradner, S., 1997, <http://www.ietf.org/rfc/rfc2119.txt>

205 **[Unicode]** *The Unicode Standard*, 5th edition, The Unicode Consortium, Addison-Wesley Professional,
206 ISBN 0321480910, <http://www.unicode.org/unicode/standard>.

207 **[UOMLSchema]** *UOML Part 1 v1.0 Schema*, [http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-part1-v1.0-](http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-part1-v1.0-schema-errata.xsd)
208 [schema-errata.xsd](http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-part1-v1.0-schema-errata.xsd)

209 **[XML1.0]** *Extensible Markup Language (XML) 1.0*, Fourth Edition. W3C. 2006.
210 <http://www.w3.org/TR/2006/REC-xml-20060816/>

211 **[XMLNamespaces]** *Namespaces in XML 1.0 (Third Edition)*. W3C. 2006. [http://www.w3.org/TR/2006/REC-](http://www.w3.org/TR/2006/REC-xml-names11-20060816/)
212 [xml-names11-20060816/](http://www.w3.org/TR/2006/REC-xml-names11-20060816/)

213 **[XMLSchema0]** *XML Schema Part 0: Primer (Second Edition)*, W3C Recommendation 28 October 2004,
214 <http://www.w3.org/TR/xmlschema-0/>

215 **[XMLSchema1]** *XML Schema Part 1: Structures (Second Edition)*, W3C Recommendation 28 October 2004,
216 <http://www.w3.org/TR/xmlschema-1/>

217 **[XMLSchema2]** *XML Schema Part 2: Datatypes (Second Edition)*, W3C Recommendation 28 October 2004,
218 <http://www.w3.org/TR/xmlschema-2/>

219

220

221 **1.8 Non-Normative References**

222 **This clause is informative.**

223 **[PDF]** ISO/IEC 32000-1, *Document Management — Portable Document Format — Part 1: PDF 1.7*.

224 **End of informative text.**

2. Abstract Document Model

UOML is based on an abstract document model. [Note: This abstract document model can describe any visual appearance; thus an arbitrary document that can be displayed and printed can be described using this abstract document model. *end note*] Description of document data using this abstract document model results in an instance of the abstract document model. An instance of the abstract document model is a hierarchy of objects, or a tree structure, on which instructions interact. This clause specifies and describes the objects of the UOML abstract document model.

2.1 Overview

In the UOML abstract document model, documents are organized hierarchically via docbase, docset and document objects (see Figure 1). There are two sub-objects of a document object: document global objects and page related objects. Document global objects include font objects. Page related objects are organized hierarchically via pages, layers, object streams, command objects and graphics objects (see Figure 2).

One docbase shall have one and only one docset, known as the root docset. The root docset is the parent of all documents, similar to the root directory of a file system. As the container for documents, docsets may be nested (i.e., a docset may be a child of another docset). Figure 1 shows how a docbase, docset and document can construct a multiple level UOML-based tree structure, similar to a file system.

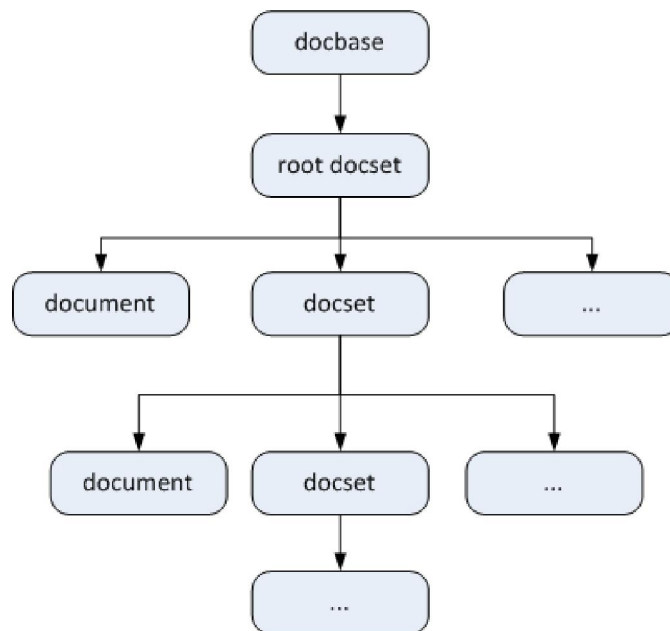


Figure1. UOML Abstract document Model 1

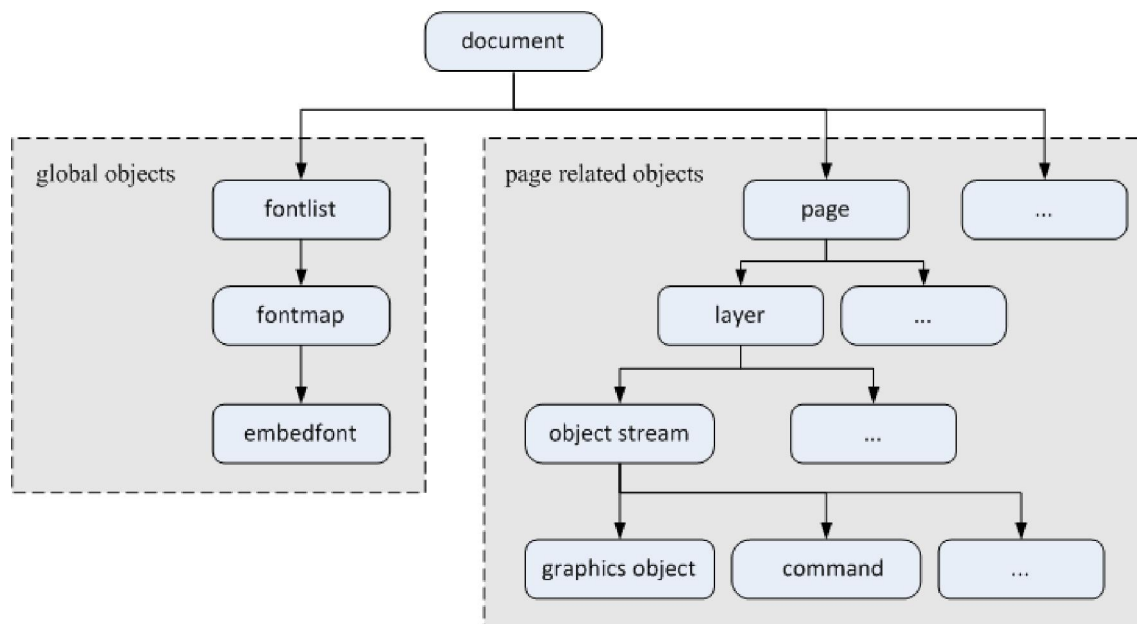


Figure 2. UOML Abstract Document Model 2

The following clauses provide a description of each object type.

2.2 Docbase

The docbase is the root of the UOML abstract document model structure. A docbase has only one docset sub-object called the root docset [*Note: Other docsets and documents are a docset's sub-objects. end note*].

The root docset is generated automatically when the docbase is created (see Figure 1). In this specification, the docbase object is specified using DOCBASE (§4.3).

Sub-object: docset.

2.3 Docset

A docset is an object whose sub-object can be a document, or another docset. In other words, a docset is a set of documents and/or docsets. In this specification, the docset object is specified using DOCSET (§4.4).

Sub-object: document, docset.

2.4 Document

The document object is the root node of document information (see Figure 2). A document contains static information for fixed-layout 2D documents [*Note: In future UOML parts or future versions of this part, other types of document information may be supported, including audio/video, 3D information, etc. end note*]. A single document has zero to multiple pages. In this specification, a document object is specified using DOC (§4.5).

[*Note: A document with no pages is permitted. It is an intermediate state. One can create such a document, then open and add pages at a future time. end note*]

266 **Sub-object:** fontlist, page.

267

268 2.5 Font

269 In the UOML abstract document model, three objects (fontlist, fontmap and embedfont), called font objects,
270 are used to describe font information used in a document. A document object may contain zero or more
271 fontlist sub-objects; a fontlist object may contain zero or more fontmap sub-objects; a fontmap may contain
272 zero or one embedfont sub-object.

273 Fontlist is a list of fontmaps. Each fontmap describes one font used in the document, including font name and
274 font sequential number used in the document. A document may optionally have font data embedded within it.

275 2.6 Page

276 A page object corresponds to a page in the document. Its sub-object is a layer object. A page object is
277 composed of zero or more layer objects. The visual appearance of a page is a combination of all layers of the
278 page.

279 Each page has its own size and resolution. The origin of a page's coordinate system is the top left corner of the
280 page. The unit of a page's logical coordinate is defined by its resolution.

281

282 In this specification, the page object is described using PAGE (§4.7).

283

284 [*Note:* A document with no pages is permitted. It is an intermediate state. One can create such a document,
285 then open and add pages at a future time. *end note*]

286

287 **Sub-object:** layer.

288 2.7 Layer

289 A layer object corresponds to one layer in a page. A layer is transparent. When a page has multiple layers, the
290 order of a layer determines the order it appears on the page, with subsequent specified layers imposed on top
291 of earlier-specified layers.

292

293 [*Note:* When a renderer processes multiple layers, the renderer processes the layers in sequence (i.e., after
294 processing all of the objects in the first layer, then move to process the objects in the second layer, and so on).
295 For example, suppose a page has 2 layers. The first layer has one object stream with three objects OA1, OA2,
296 OA3, and the second layer has one object stream with two objects OB1, OB2. The renderer should treat the
297 rendering result as a Layer with an object stream containing objects OA1, OA2, OA3, OB1, and OB2 in sequence.
298 In summary, the layers should be treated as one layer containing all of the graphics objects and command
299 objects in sequence. There is no particular blending effect between layers. Any overlapping effect is controlled
300 by command object with type ROP (Raster_OP), which will change the current graphics state of ROP. *end note*]

301

302 In this specification, the layer object is described using LAYER.

303

304 **Sub-object:** object stream.

2.8 Object Stream

An object stream is a sequence of zero or more graphics objects and/or command objects.

A layer holds 0 or more object streams. The reason a layer can hold many object streams is that multiple object streams may be needed to specify a related set of graphics and command objects, each of which is combined in one layer. The different object streams can then be handled separately; for example, for future extensions for such functionality as security control.

Sub-object: graphics object, command object.

2.9 Graphics Object

A graphics object is a set of objects that could allow the render engine to draw text, image, and Path. Graphics objects describe the appearance of the page. The graphics objects in UOML includes arc, Bezier, circle, ellipse, image, line, rectangle, round rectangle, Path and text objects.

2.10 Command Object

A command object changes one or multiple parameters in the current graphics state. The graphics state is initialized at the beginning of the rendering of each layer with the default values specified in section §4.13. The rendering of a graphics object relies on the current parameters in the graphics state.

2.11 UML Diagram of UOML

The following is a UML diagram of the UOML abstract document model. It shows the tree structure of UOML along with cardinalities associated with the objects discussed in this clause.

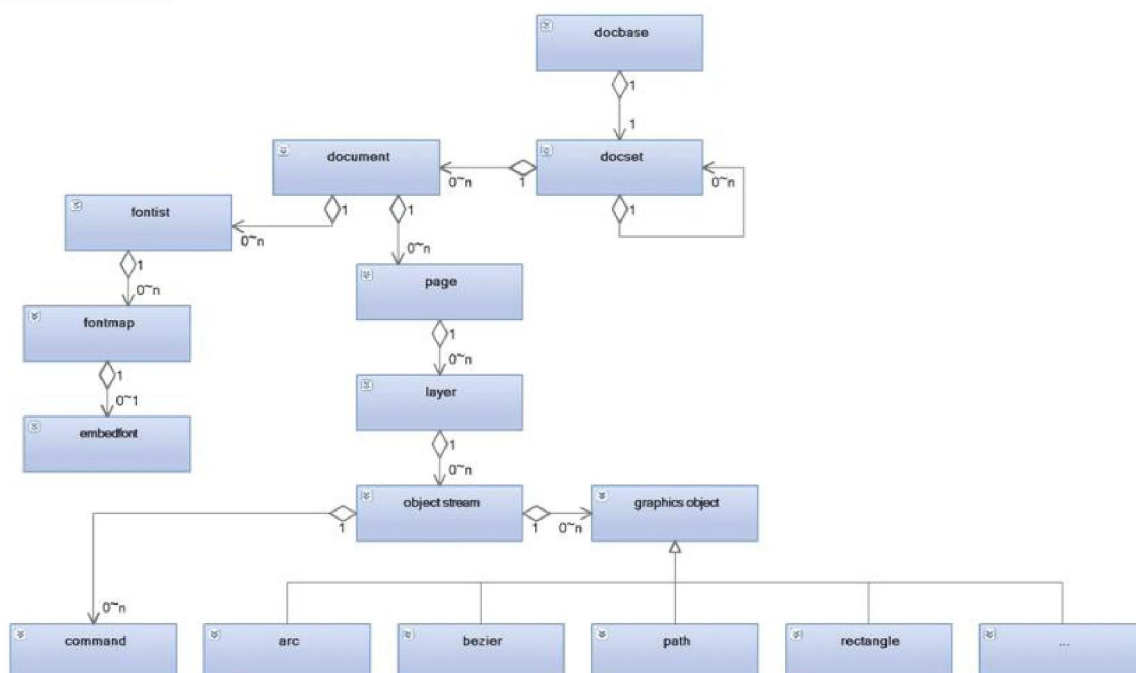


Figure 3. UML diagram of UOML abstract document model

2.12 Page Rendering Model

The following are the steps to render a page:

1. Repeat the following step from the first layer to the last layer by position number.
 - a. Initialize the current graphics state of the rendering engine with the default value (§4.13).
 - b. Loop through the object streams of the current layer by position number.
 - i. Then loop through the objects of each object stream by position number.
 1. Draw the object if it is a graphics object.
 2. Otherwise, the object is a command object; update the graphics state according to the object.
2. Page rendering completes.

3. UOML Instructions

UOML Instructions are used to define operations that interact with UOML objects, such as creating a docbase, inserting a sub-object, deleting an object, changing an attribute of an object, etc.

This clause defines the syntax and semantics of the UOML instructions. The order of UOML instructions are OPEN, followed by zero or many operations except OPEN or CLOSE, ended by CLOSE. There are no dependencies among operations between OPEN and CLOSE; thus there is no order for those operations.

3.1 OPEN

Semantics:

OPEN creates or opens a docbase.

Properties:

create: a Boolean value representing whether to create a docbase if it does not exist. Specifying 'true' will create the docbase. The default value is 'true'.

del_exist: a Boolean value, representing whether to delete the docbase if it already exists. Specifying 'true' deletes the existing docbase. The default value is 'false'.

path: a character string value, representing the location of a docbase. There is no defined format for the path value (e.g., URI, URL, fully-qualified file system directory path, absolute value, relative value, etc.). Valid values for this property, and their appropriate interpretation, are implementation-defined. [Note: A path should be a format such that it could be used to find the location of the docbase. *end note*]

Sub-elements: N/A

Return value:

If OPEN succeeds, the returned RET element contains a 'stringVal' sub-element with the 'name' property as the handle and the 'val' property represents the handle of the docbase. [Note: The syntax of the handle value is implementation-defined and has no relationship to other handles returned by the given DCMS nor to other handles returned by another DCMS, even for the creation of the same document. *end note*]

If OPEN fails, the return value is defined by RET (§3.9).

[Example:

Create a docbase, named 1.sep. If the DCMS successfully processed the OPEN instruction, it will return a RET instruction.

```
<uoml:OPEN path="/home/admin/storage/1.sep" create="true" del_exist="false"/>
```

Return element if OPEN succeeds:

```
<uoml:RET>
```

```
  <boolVal name="SUCCESS" val="true"/>
```

```
  <stringVal name="HANDLE" val="db_handle_xxxxx"/>
```

```

380         </uoml:RET>
381
382         Return element if OPEN fails:
383
384         <uoml:RET>
385             <boolVal name="SUCCCESS" val="false"/>
386             <stringVal name="ERR_INFO" val="required resource not available"/>
387         </uoml:RET>
388
389     end example]

```

390 3.2 CLOSE

391 Semantics:

392 CLOSE closes a docbase

393 Properties:

394 *handle*: a character string value, representing the handle of the docbase to be closed.

395 Sub-elements: N/A

396 Return value:

397 Defined by RET

398 [Example:

399 Close a docbase.

```

400
401         <uoml:CLOSE handle="db_handle_xxxxx"/>
402
403     end example]

```

404 3.3 USE

405 Semantics:

406 USE sets an object as the current object. [Note: USE sets an object in the document to the current
407 object of focus. The current object is used when the destination object is not specified within an
408 instruction (e.g. INSERT). end note]

409 Properties:

410 *handle*: a character string value, representing the handle of current object to be set up.

411 Sub-elements: N/A

412 Return value:

413 Defined by RET

414 [Example:

415 Set up the handle represented object as the current object.

```

416
417         <uoml:USE handle="obj_handle_xxxxxx"/>
418
419     end example]

```


3.4 GET

Semantics:

GET retrieves information such as a sub-object handle, the count of sub-objects, the property value of an object, or a page bitmap.

Properties:

usage: a character string value, representing the usage of GET. The possible values of this property are GET_SUB, GET_SUB_COUNT, GET_PROP, GET_PAGE_BMP, representing getting a sub-object, getting the sub-object count, getting properties, and getting a page bitmap, respectively.

handle: a character string value, representing the object handle of the current operation. This property is optional. If this property is not used, then the current handle set by the USE instruction is used.

Sub-elements:

pos: used when usage=GET_SUB.

Property of this sub-element:

val: specifies the position number of the specified sub-object, starting from 0.

Sub-element of this sub-element: N/A

property: used when usage=GET_PROP.

Property of this sub-element:

name: specifies the name of the property whose value is returned, if *name* is an empty string, the type of the object is retrieved.

Sub-element of this sub-element: N/A

disp_conf: used when usage=GET_PAGE_BMP.

Properties of this sub-element:

end_layer: specifies the handle of the end layer of the operation (the drawing operation ends at this layer and this layer is not drawn any more)

resolution: represents resolution of bitmap

format: represents the bitmap format. The only valid value is "bmp", representing the uncompressed BMP format.

output: represents whether to put out to the file or to the memory. Possible values for this property are FILE or MEMORY;

addr: represents the path of output file or memory address.

Sub-element of this sub-element:

clip: represents clip area for output, PATH type.

Usage value / Return value:

The return value is based on the usage value:

- GET_SUB_COUNT: If the usage is GET_SUB_COUNT, this indicates to get the number of sub-objects of this specific object. In this case, there is no sub-element needed for the GET instruction. The return value, which is returned via the RET instruction, contains one 'intVal' sub-element. Its 'name' property is "sub_count" and the 'val' property represents number of sub-objects.

[Example:

Get the total number of sub-objects of the specific object:

```
<uoml:GET handle="obj_handle_xxx" usage="GET_SUB_COUNT"/>
```

RET instruction returns the number:

```
<uoml:RET >
  <boolVal name="SUCCESS" val="true"/>
  <intVal name="sub_count" val="1"/>
</uoml:RET>
```

end example]

- GET_SUB: If the usage is GET_SUB, this indicates to get the handle of some specific sub-object. In this case, GET shall contain the sub-element of 'pos'. The return value, which is returned via the RET instruction, contains one 'stringVal' sub-element. Its 'name' property is "handle" and its 'val' property represents the sub-object's handle.

[Example:

Get a specific sub-object handle:

```
<uoml:GET handle="obj_handle_page01" usage="GET_SUB">
  <pos val="0"/>
</uoml:GET>
```

RET instruction returns the handle of the sub-object:

```
<uoml:RET>
  <boolVal name="SUCCESS" val="true"/>
  <stringVal name="handle" val="obj_handle_layer01"/>
</uoml:RET>
```

end example]

- GET_PROP: If the usage is GET_PROP, this indicates to get some specific property of a specific object. If the name property is a non-empty string, GET shall contain the sub-element of 'property'. If the operation succeeds, the sub-element of return value, which is returned via RET instruction, is variant; the sub-element name relies on the type it has retrieved, the 'name' property of the sub-element is the property name to get, 'val' property is the value of the property; otherwise if the name property is an empty string, the RET instruction returns a stringVal value representing the type of the object, which is the element name of the XML

description of the object without the namespace prefix.

[Example:

Get specific property of the object

```
<uoml:GET handle="obj_handle_xxxxx" usage="GET_PROP">
  <property name="start"/>
</uoml:GET>
```

RET instruction returns the start property, which is a coordinate:

```
<uoml:RET>
  <boolVal name="SUCCESS" val="true"/>
  <stringVal name="start" val="200,300"/>
</uoml:RET>
```

end example]

- GET_PAGE_BMP: If the usage is GET_PAGE_BMP, this indicates to get the specific page bitmap. In this case, GET shall contain the sub-element 'disp_conf'. The requested bitmap should be placed/returned where the 'addr' and 'output' property of the 'disp_conf' element is specified.

[Example:

Get specific page's bitmap

```
<uoml:GET handle="page_obj_handle_xxx" usage="GET_PAGE_BMP">
  <disp_conf format="bmp" output="FILE" end_layer="1" resolution="600"
    path="/home/admin/output/page.bmp">
    <clip>
      <subpath data="s 0,0 1 3000,0 1 3000, 5000 1 0, 5000 1 0,0"/>
    </clip>
  </disp_conf>
</uoml:GET>
```

end example]

- When GET fails, the return value is defined by RET.

[Example:

```
<uoml:RET>
  <boolVal name="SUCCESS" val="false"/>
  <stringVal name="ERR_INFO" val="disk full"/>
</uoml:RET>
```

end example]

551 3.5 SET

552 Semantics:

553 Set property values for an object. It may contain one or more sub-element(s).

554 The 'name' property of the sub-element represents which property of specific object will be modified.

555 The 'val' property of the sub-element contains the new property value.

556 Properties:

557 *handle*: a character string value, representing the handle of which property value needs to be modified.

558 This property is optional. If this property is not used, then use the handle set from USE instead.

559 Sub-element:

560 *intVal*: set up integer type value, INT type

561 *floatVal*: set up float type value, DOUBLE type.

562 *timeVal*: set up time value, TIME type.

563 *dateVal*: set up date value, DATE type.

564 *dateTimeVal*: set up date and time value, DATETIME type.

565 *durationVal*: set up time duration value, DURATION type.

566 *stringVal*: set up string type value, STRING type.

567 *binaryVal*: set up binary type value, BINARY type.

568 *compoundVal*: set up compound type value, COMPOUND type.

569 *boolVal*: set up boolean type value, BOOLEAN type.

570 Return value:

571 defined by RET.

572 [Example:

573 Set specific object's angle property.

574 `<uoml:SET handle="obj_handle_XXXXXX">`

575 `<floatVal name="angle" val="0.1"/>`

576 `</uoml:SET>`

577 *end example]*

578

579 3.6 INSERT

580 Semantics:

581 INSERT inserts an object as a sub-object of a specific parent object.

582 Properties:

583 *handle*: a character string value, representing the handle of parent object. This property is optional. If

584 this property is not used, then use the handle set from USE instead.

pos: int value, starting from 0, representing the insert location. The object shall be inserted before the object at *pos*. This property is optional. If this property is not used, insert after the last sub-object. If *pos* is greater than or equal to the number of items in the sequence then the insertion point is implementation-defined. After the insertion, the position numbers of all items after the inserted item are increased by one.

Sub-element:

xobj: xml expression of the sub-object.

Return value:

If the insertion succeeds, RET shall contain one sub-element 'stringVal'. Its 'name' property is handle and its 'val' property represents the handle of the newly inserted sub-object.

[Example:

Insert text data

```
<uoml:INSERT pos="1"/>
  <xobj>
    <text origin="100, 200" encode="ASCII" text="UOML"
      spaces="20,20,20"/>
  </xobj>
</uoml:INSERT>
```

end example]

[Example:

Insert a layer

```
<uoml:INSERT handle="page_obj_handle_XXXXXX">
  <xobj>
    <layer/>
  </xobj>
</uoml:INSERT>
```

end example]

3.7 DELETE

Semantics:

DELETE deletes an object. After a deletion, the position numbers of all items after the deleted item are decreased by one. [*Note*: In other words, the range of items should not include any empty position spots. *end note*]

Properties:

handle: a character string value, representing the object to be deleted. This property is optional. If this property is not used, then use the handle set from USE instead.

Sub-element: N/A

629 **Return value:**

630 Defined by RET

631 *[Example:*

632 Delete an object

633

634 `<uoml:DELETE handle="img_obj_handle_xxx"/>`

635 *end example]*

636

637 3.8 SYSTEM

638 **Semantics:**

639 SYSTEM executes system maintenance, such as saving the docbase. [*Note:* Within this Part of the UOML
640 specification, SYSTEM has only one function: to save the docbase. *end note*]

641 **Properties:**

642 N/A

643 **Sub-element:**

644 *flush*: the 'handle' property of this sub-element represents the handle of a docbase object, and the
645 'path' property represents the saving path for the docbase.

646 **Return value:**

647 Defined by RET

648 *[Example:*

649 Save the docbase example.sep

650

651 `<uoml:SYSTEM>`
652 `< flush handle="docbase_handle_xxxxx"`
653 `path="/home/admin/storage/example.sep"/>`
654 `</uoml:SYSTEM>`

655 *end example]*

656

657 3.9 RET

658 **Semantics:**

659 RET is the return value from the DCMS to the application software. RET may contain one or more
660 return values, and each return value is represented by one sub-element (e.g., boolVal, stringVal, intVal,
661 floatVal, compoundVal, etc.).

662 The 'name' property of the sub-element represents the name of the return value.

663 If the return value is a simple type, the 'val' property of sub-element contains the return value.

664 If the return value is a compound type, a sub-element will be added under the corresponding sub-
665 element to represent the compound return value.

666 RET contains at least one 'boolVal' sub-element to describe whether the operation was successful or

667 not. Its 'name' property is SUCCESS, and its 'val' property is either 'true' or 'false', depending on the
668 success of the operation.

669 When the operation fails, RET also contains one 'stringVal' sub-element. Its 'name' property is
670 ERR_INFO, and its 'val' property describes the failure information, in an implementation-defined way.
671 [Note: For other return values, check the definition of the concrete UOML instruction for reference. *end*
672 *note*]

673 [Example: <boolVal name="SUCCESS" val="true"/> *end example*]

674

675 **Properties:** N/A

676

677 **Sub-element:**

678 *intVal*: integer type return value, INT type

679 *floatVal*: float type return value, DOUBLE type.

680 *TimeVal*: time type return value, TIME type.

681 *DateVal*: date type return value, DATE type.

682 *DateTimeVal*: date and time type return value, DATETIME type.

683 *DurationVal*: time duration type return value, DURATION type.

684 *StringVal*: string type return value, STRING type.

685 *BinaryVal*: binary type return value, BINARY type.

686 *CompoundVal*: compound type return value, COMPOUND type.

687 *BoolVal*: boolean type return value, BOOLEAN type.

688

689 [Example:

690 Return two values.

691 <uoml:RET>

692 <boolVal name="SUCCESS" val="false"/>

693 <stringVal name="ERR_INFO" val="required resource not available"/>

694 </uoml:RET>

695 *end example*]

696

4. UOML Objects

697

This clause describes the objects defined by the UOML abstract document model. The description shows the XML representation of each object. These objects are used as part of the UOML instructions.

698

699

The formal definitions of the XML vocabulary for these objects are specified in the UOML XML Schema Definition located at [UOMLSchema].

700

701

702

4.1 Logical Coordinate System and Units

703

A UOML document uses a logical coordinate system. The terms *position*, *point* and *coordinate* may be used interchangeably. They refer to a logical point in the logical coordinate system. The origin of the logical coordinate system is the top left point. The direction of the x-axis is left to right. The direction of the y-axis is top to bottom.

704

705

706

707

708

The length of the units along each axis depends on the resolution property of the page. If the resolution of a page is x , the length of the unit along each axis is $2.54/x$ cm. A logical unit indicates one inch divided by the resolution of the page.

709

710

711

712

The resolution of each page is the same along the x and y axis.

713

714

UOML uses radians as the unit of measurement for angles. [*Note*: Though different from PDF, XSL-FO and SVG, conversion can be easily made without any loss of information. *end note*]

715

716

4.2 Graphics State

717

A DCMS shall maintain an internal data structure called the *graphics state* that holds the current graphics control parameters. The graphics state is initialized at the beginning of each layer with the default values specified in section §4.13. The rendering of a graphics object relies on the current parameters in the graphics state. A command object changes one or many parameters in the current graphics state.

718

719

720

721

4.3 DOCBASE

722

Semantics: XML representation of the docbase object (§2.2).

723

Properties:

724

name: name of docbase.

725

path: specifies the location of the docbase. *path* is readonly. Its value is the same value of the 'path' property of OPEN when this docbase was created.

726

727

Sub-elements: N/A

728 4.4 DOCSET

729 **Semantics:** XML representation of the docset object (§2.3).

730 **Properties:**

731 *name*: name of docset.

732 **Sub-elements:** N/A

733 4.5 DOC

734 **Semantics:** XML representation of the document object (§2.4).

735 **Properties:**

736 *name*: name of document.

737 **Sub-elements:**

738 *metainfo*: metadata of the document, METALIST type.

739

740 4.5.1 Metadata

741 General information, such as the document's title, author, creation and modification date, is called metadata.
742 Metadata is defined using keys and values. [*Note*: A key is not necessarily unique. A detailed specification of
743 the keys and value falls outside the scope of this specification. *end note*]. In this specification, metadata is
744 described using METALIST and META.

745 4.5.1.1 METALIST

746 **Semantics:** A list of all the metadata in the document.

747 **Properties:** N/A

748 **Sub-elements:**

749 *meta*: META type.

750 4.5.1.2 META

751 **Semantics:** One item of metadata.

752 **Properties:**

753 *key*: character string value representing the key of metadata. [*Note*: A key is not necessarily unique. A
754 detailed specification of the keys and value falls outside the scope of this specification. *end note*]

755 *val*: character string value representing the value of metadata.

756 **Sub-elements:** N/A

757

758 4.6 FONT DEFINITION

759 Fontlist, fontmap and embedfont are called font objects. This clause gives the XML description of these objects.

760 4.6.1 FONTLIST

761 **Semantics:** A list of all the fonts used in the document. It is the XML description of the fontlist object (§2.5).

762 **Properties:** N/A

763 **Sub-elements:** N/A

764 4.6.2 FONTMAP

765 **Semantics:** Defines one font used in the document. It is the XML description of the fontmap object (§2.5).

766 **Properties:**

767 *name*: name of the font

768 *no*: non-negative integer value representing the id of the font quoted in document *no* is used for fast
769 quoting. If its value is zero, the font need not be fast quoted. If its value is non-zero, the result is unique
770 within the scope of the document.

771 **Sub-elements:** N/A

772 4.6.3 EMBEDFONT

773 **Semantics:** Defines one embedded font type. It is the XML description of the embedfont object (§2.5). Use
774 OpenFont as an embedded font type. After encoding OpenFont using base64 format, put the result into
775 EMBEDFONT's content section as the embedded font data.

776 **Properties:** N/A

777 **Sub-elements:** N/A

778 4.7 PAGE

779 **Semantics:** XML description of the page object (§2.6).

780 **Properties:**

781 *width*: positive float value representing the width of the page in pixels.

782 *height*: positive float value representing the height of the page in pixels.

783 *resolution*: positive integer value representing the resolution of the page, which defines the unit of a
784 pixel (§4.1).

785 **Sub-elements:** N/A

786 4.8 LAYER

787 **Semantics:** XML description of the layer object (§2.7).

788 **Properties:** N/A

789 **Sub-elements:** N/A

790 4.9 OBJSTREAM

791 **Semantics:** XML description of the object stream object (§2.8).

792 **Properties:** N/A

793 **Sub-elements:** N/A

794 4.10 Graphics Objects

795 Graphics objects describe the appearance of the page. The following clauses gives the XML description of each
796 graphics object.

797

798

799 4.10.1 ARC

800 **Semantics:**

801 An arc of an ellipse, specified by a starting, ending, and center position, along with a direction and
802 angle.

803 **Properties:**

804 *start*: starting position of the arc.

805 *end*: ending position of the arc.

806 *center*: center of the arc's ellipse.

807 *clockwise*: the direction for arc is from the starting point to the ending point, which can be clockwise or
808 counterclockwise. As a Boolean value, "true" represents clockwise and "false" represents
809 counterclockwise.

810 *angle*: inclination from coordinate system's x-axis to arc's x-axis. It is specified using a radian value. A
811 positive value represents counterclockwise and a negative value represents clockwise.

812 **Sub-elements:** N/A

813 4.10.2 BEZIER

814 **Semantics:**

815 A second-order or third-order Bezier curve. A Bezier curve is specified using three or four properties:
816 the starting point, the ending point, one control point and, optionally, a second control point. A
817 second-order Bezier curve is specified when only one control point is used. A third-order Bezier curve is
818 specified when a second control point is used.

819 **Properties:**

820 *start*: starting point of the Bezier curve.

821 *ctrl*: the first control point of the Bezier curve.

822 *ctrl2*: the optional second control point of the Bezier curve.

823 *end*: ending point of the Bezier curve.

824 **Sub-elements:** N/A

825 4.10.3 CIRCLE

826 Semantics:

827 A circle, specified by a center and radius.

828 Properties:

829 *center*: coordinate of the circle center.

830 *radius*: positive integer value representing the radius of the circle.

831 Sub-elements: N/A

832 4.10.4 ELLIPSE

833 Semantics:

834 An ellipse, specified by a center, x and y radius, and a rotation angle.

835 Properties:

836 *center*: coordinates of ellipse center.

837 *xr*: positive integer value representing the length of the x-radius.

838 *yr*: positive integer value representing the length of the y-radius.

839 *angle*: inclination from coordinate system's x-axis to ellipse's x-axis. It is specified using a radian value
840 of type xs:float. A positive value represents counterclockwise and a negative value represents clockwise.

841 Sub-elements: N/A

842 4.10.5 IMAGE

843 Semantics:

844 An image, specified by top-left and bottom-right corner coordinates, the image type, and either the
845 image location or the image content. The intrinsic image aspect ratio may be different than the aspect
846 ratio of the box described by the two corners; in this case, the image should be stretched to fit the box
847 described by the two corners. [Note: An image may contain a large amount of data, and parsing this
848 data may greatly reduce the performance of an XML processor. It is recommended to specify large
849 images using a file and its location. *end note*]

850 Properties:

851 *tl*: coordinates of the top-left corner of the image

852 *br*: coordinates of the bottom-right corner of the image

853 *type*: image type, possible values include "bmp", "png", "jpeg", "jbig", "tiff", representing BMP, PNG,
854 JPEG, JBIG, TIFF images respectively.

855 *path*: path of the image file. This is an optional property, but if present, the content of IMAGE element
856 should be left blank; otherwise the content of IMAGE element contains the base64 encoded raw image
857 data.

858 Sub-elements: N/A

859 **Sub-objects:** N/A

860 **4.10.6 LINE**

861 **Semantics:**

862 A line, specified by a starting and ending point.

863 **Properties:**

864 *start*: coordinates of where the line starts.

865 *end*: coordinates of where the line ends.

866 **Sub-elements:** N/A

867

868 **4.10.7 RECT**

869 **Semantics:**

870 A rectangle, specified by the coordinates of the top-left and bottom-right corner.

871 **Properties:**

872 *tl*: coordinates of the top-left corner of the rectangle.

873 *br*: coordinates of the bottom-right corner of the rectangle.

874 **Sub-elements:** N/A

875

876 **4.10.8 ROUNDRECT**

877 **Semantics:**

878 A rectangle with round corners. The round corner of a round rectangle is a quarter of an ellipse.

879 **Properties:**

880 *tl*: coordinates of the top-left corner of the rectangle.

881 *br*: coordinates of the bottom-right corner of the rectangle.

882 *xr*: positive integer value representing the x-radius of the round corner.

883 *yr*: positive integer value representing the y-radius of the round corner.

884 **Sub-elements:** N/A

885 **4.10.9 SUBPATH**

886 **Semantics:**

887 A subpath specifies a chain of curves consisting of lines, Bezier curves and arcs. It can be either closed
888 or open.

889 **Properties:**

890 *data*: specifies the ordered set of graphics objects describing the subpath from the starting point of
891 the first object, through each of the subsequent objects, to the ending point of the last object. It is an
892 ordered set of operands and coordinate arguments for each operand expressed in a single string value.
893 [*Note*: Refer to §4.11.12 for the encoding of property data. *end note*]

894 **Sub-elements:** N/A

895 [Example: The following example demonstrates inserting of a Path object using INSERT instruction. The Path
896 consists of two subpaths: a rectangle formed by four straight lines, and a curved line segment formed by Bezier
897 curves.

```
898      <uoml:INSERT pos="2" handle="vs03">  
899      <xobj>  
900      <path>  
901      <subpath data="s 214,193 l 368,193 l 368,298 l 214,298"/>  
902      <subpath data="s 417,206 B 417,186 426,167 435,167 B 443,167 452,230 452,293"/>  
903      </path>  
904      </xobj>  
905      </uoml:INSERT>
```

906
907 *end example*].

908

909 4.10.10 PATH

910 Semantics:

911 A Path specifies an open or closed region consisting of a collection of one or many subpaths, circles,
912 ellipses, rectangles and round rectangles expressed using sub-elements. The PATH element itself does
913 not contain any properties or data.

914 **Properties:** N/A

915 Sub-elements:

916 *circle*: CIRCLE type, defines a circle.

917 *ellipse*: ELLIPSE type, defines an ellipse.

918 *rect*: RECT type, defines a rectangle.

919 *roundrect*: ROUNDRECT type, defines a rectangle with round corners.

920 *subpath*: SUBPATH type, defines a subpath.

921

922 [Example: The following example demonstrates a PATH consisting of two sub elements: a rectangle and a
923 circle.

```
924      <uoml:INSERT pos="4">  
925      <xobj>  
926      <path>  
927      <circle center="167,251" radius="70" />  
928      <rect tl="124,135" br="345,257"/>  
929      </path>  
930      </xobj>  
931      </uoml:INSERT>
```

932

933 *end example*].

934

935 4.10.11 TEXT

936 Semantics:

937 Text, specified using an origin, encoding information, text data and an optional character spacing list.

938 Properties:

939 *origin*: the coordinate of the first character's origin. The origin of a character is defined by its font
940 information.

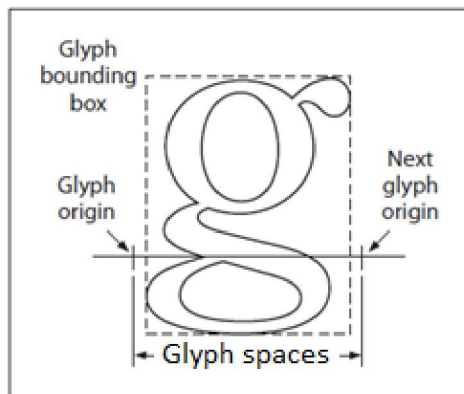
941 *encode*: character set or encoding of text data. The valid value for this property should be one of the
942 character encodings registered (as charsets) with the Internet Assigned Numbers Authority [IANA-
943 CHARSETS], otherwise it should use names starting with an x- prefix.

944 *text*: character data contained in text, base64 encoded string data.

945 *spaces*: an optional, ordered set of distances that specifies distances between adjacent characters'
946 origins, separated by a comma.

947

948 The origin of a character refers to the point (0, 0) in the coordinate system of the character glyph, as
949 illustrated in the Figure 4. When a text object with only one character is specified and the text object
950 has coordinate (x, y), the rendering engine should place the origin of the character at (x, y) and render
951 the character.



952

953 Figure 4. spaces of text

954 The spaces property is the offset or distance between the x coordinates of two adjacent characters. It is
955 always positive. The number of comma-separated values shall be one fewer than the number of
956 characters in the string. The values should override the widths of the characters as specified by the font
957 used. The values are used to calculate the coordinate to place the origin of each character.

958

959 Sub-elements: N/A

960

961 4.10.12 Coordinate and subpath Encoding Rules

962 In order to provide short and efficient expression for coordinates and Path, this section defines the encoding
963 rules used by UOML.

964

965 **Coordinate encoding rules**

966
967 `coord = coordx, [blank] , ',' , [blank] , coordy ;`
968 `coordx = number ;`
969 `coordy = number ;`

970

971 In this Backus-Naur Form rule expression, "coord" are coordinates, "coordx" is coordinate x, "coordy"
972 is coordinate y, and "number" represents a string form of an integer number.

973

974 **Path encoding rules**

975

976

977 `path = start , { blank , (line | bezier2 | bezier3 | arc) } ;`
978 `start = 's' , blank , coord ;`
979 `line = 'l' , blank , coord ;`
980 `bezier2 = 'b' , blank , coord , blank , coord ;`
981 `bezier3 = 'B' , blank , coord , blank , coord , blank , coord ;`
982 `arc = 'a' , blank , clockwise , blank , angle , blank , coord , blank , coord ;`
983 `clockwise = 'true' | 'false' ;`
984 `angle = float ;`
985 `number = ['-'] , digit , { digit } ;`
986 `float = number [, '.' , { digit }][, ('e' | 'E') , ['+' | '-'] , digit , {digit}] ;`
987 `digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;`
988 `blank = ' ' , { ' ' } ;`

989

990 **Semantics**

991 "coord" represents coordinates.

992 "start" represents the start point of the subpath.

993 "line" represents a line segment.

994 "bezier2" represents a second-order Bezier curve.

995 "bezier3" represents third-order Bezier curve.

996 "blank" represents one or many blanks or an equivalent whitespace character, such as a tab, carriage
997 return or a new line.

998

999 In the definition of "line", the "coord" represents the ending point.

1000 In the definition of "bezier", the two "coord" are for the control point and the ending point.

1001 In the definition of "bezier3", the three "coord" are for the control point 1, control point 2 and
1002 ending point.

1003 In the definition of "arc", the two "coord" are the center and end points.

1004 [Note: The start point of each item is the previous end point. *end note*]

1005

1006 4.11 Command Object

1007 A command object is used for modifying the graphics, such as text size, typeface and color.

1008 4.11.1 CMD

1009 **Semantics:** XML description of command objects.

1010 **Properties:**

1011 name: name of the command. [*Note: §4.12.2 provides possible values for this property. end note*]

1012 v1: optional command value.

1013 v2: optional command value.

1014 **Sub-elements:**

1015 *rgb*: a COLOR_RGB value (§4.11.3.1), used when 'name' is one of COLOR_LINE, COLOR_FILL,
1016 COLOR_SHADOW, COLOR_OUTLINE or COLOR_TEXT.

1017 *matrix*: a MATRIX value (§4.11.3.2), used when 'name' is one of TEXT_MATRIX, IMAGE_MATRIX,
1018 GRAPH_MATRIX or EXT_MATRIX.

1019 *cliparea*: a PATH value, used when 'name' is CLIP_AREA.

1020 **Sub-objects:** N/A

1021 [*Example:*

```
1022
1023 <uoml:INSERT pos="2" handle="vs03">
1024   <xobj>
1025     <cmd name="COLOR_LINE" >
1026       <rgb r="128" g="3" b="255" a="120"/>
1027     </cmd>
1028   </xobj>
1029 </uoml:INSERT>
```

1031 *end example]*

1033 [*Example:*

```
1034
1035 <uoml:INSERT pos="2" handle="vs03">
1036   <xobj>
1037     <cmd name="LINE_CAP" v1="END_BUT"/>
1038   </xobj>
1039 </uoml:INSERT>
```

1041 *end example]*

1043 [*Example:*

```
1044 <uoml:INSERT pos="2" handle="vs03">
```

```

1045     <xobj>
1046         <cmd name="TEXT_MATRIX">
1047             <matrix f11="2" f12="0" f21="0" f22="1.5" f31="10" f32="20"/>
1048         </cmd>
1049     </xobj>
1050 </uoml:INSERT>

```

1051

1052 *end example]*

1053 4.11.2 Values for CMD's 'name' property

1054 This clause describes the values that may be used for CMD's 'name' property, and which properties and sub-
1055 elements may be used for each valid 'name' value. [*Example:* If the CMD's 'name' property is 'COLOR_LINE',
1056 then CMD's sub-element is 'rgb'. *end example]*

1057

1058 In order to simplify the parsing process, properties (command values) within command objects all have a
1059 general name called v1 (and v2 if there is a second property) no matter what they represent.

1060 4.11.2.1 COLOR_LINE

1061 **Semantics:** Set the current line color

1062 **Properties:** N/A

1063 **Sub-elements:**

1064 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to stroke lines and
1065 curves.

1066 4.11.2.2 COLOR_FILL

1067 **Semantics:** Set the current fill color

1068 **Properties:** N/A

1069 **Sub-elements:**

1070 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to fill an area.

1071 4.11.2.3 COLOR_SHADOW

1072 **Semantics:** Set the current character shadow color

1073 **Properties:** N/A

1074 **Sub-elements:**

1075 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to draw the shadow of
1076 characters.

1077 4.11.2.4 COLOR_OUTLINE

1078 **Semantics:** Set the current character outline color

1079 **Properties:** N/A

1080 **Sub-elements:**

1081 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to draw the outline of
1082 characters.

1083 4.11.2.5 COLOR_TEXT

1084 **Semantics:** Set the current text color

1085 **Properties:** N/A

1086 **Sub-elements:**

1087 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to draw characters.

1088 4.11.2.6 LINE_WIDTH

1089 **Semantics:** set the current line width/thickness

1090 **Properties:**

1091 *v1*: a positive floating point number, representing the width of the line.

1092 **Sub-elements:** N/A

1093 4.11.2.7 LINE_CAP

1094 **Semantics:** Set the current line cap style

1095 **Properties:**

1096 *v1*: a character string, representing the line cap style. Possible values for this property are END_BUT,
1097 END_ROUND and END_SQUARE.

1098 END_BUT: the stroke shall be squared off at the endpoint of the path. There shall be no projection
1099 beyond the end of the path.



1100

1101

1102 END_ROUND: a semicircular arc with a diameter equal to the line width shall be drawn around the end
1103 point the endpoint and shall be filled in.



1104

1105

1106 END_SQUARE: the stroke shall continue beyond the endpoint of the path for a distance equal to half
1107 the line width and shall be squared off.



1108

1109

1110 **Sub-elements:** N/A

1111 4.11.2.8 LINE_JOIN

1112 **Semantics:** Set the current line join style

1113 **Properties:**

1114 v1: a character string, representing the line join style. Possible values for this property are JOIN_MITER,
1115 JOIN_BEVEL and JOIN_ROUND

1116

1117 JOIN_MITER: the outer edges of the strokes for the two segments shall be extended until they meet at
1118 an angle. If the segments meet at too sharp an angle as measured by the current miter length
1119 maximum, the value JOIN_BEVEL shall be used instead.



1120

1121

1122 JOIN_BEVEL: the two segments shall be finished with END_BUT and the resulting notch beyond the end
1123 of the segments shall be filled with a triangle.



1124

1125

1126 JOIN_ROUND: an arc of a circle with a diameter equal to the line width shall be drawn around the point
1127 where the two segments meet, connecting the outer edges of the strokes for the two segments. This
1128 pie slice-shaped figure shall be filled in, producing a rounded corner.



1129

1130

1131 **Sub-elements:** N/A

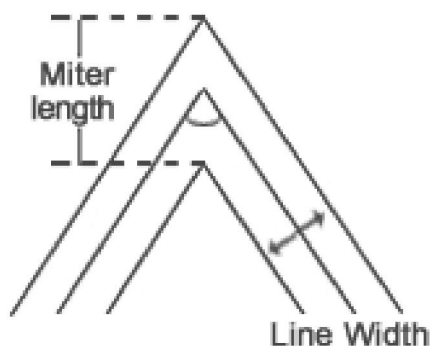
1132 4.11.2.9 MITER_LIMIT

1133 **Semantics:** Impose a maximum on the ratio of the miter length to the line width. When the limit is exceeded,
1134 the join is converted from a miter to a bevel.

1135

1136 **Properties:**
1137 v1: a positive floating point number, representing the maximum ratio.

1138 **Sub-elements:** N/A
1139

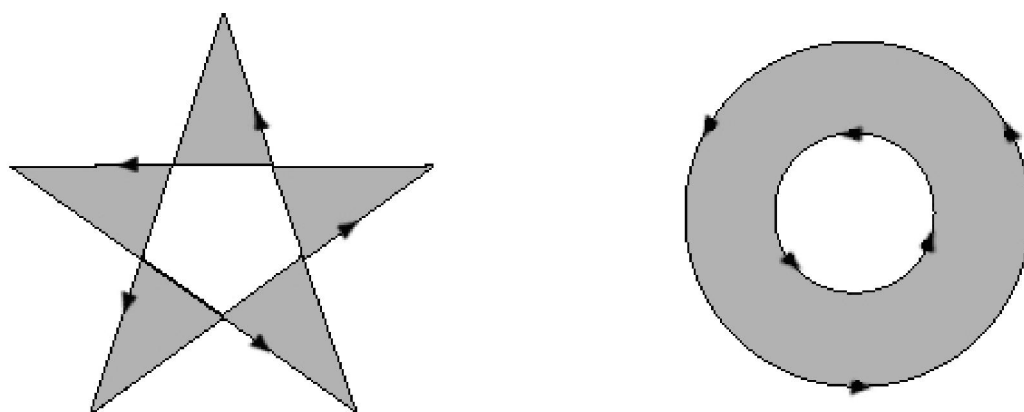


1140
1141 **4.11.2.10 FILL_RULE**

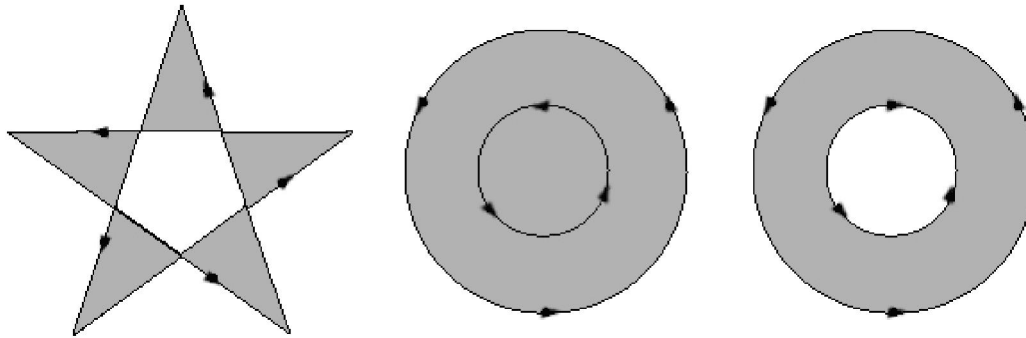
1142 **Semantics:** Set the current fill rules

1143 **Properties:**
1144 v1: a character string, representing the fill rule. The possible values for this property are
1145 RULE_EVENODD and RULE_WINDING.

1146
1147 **RULE_EVENODD:** Specifies that areas are filled according to the even-odd parity rule. According to this
1148 rule, it can be determined whether a test point is inside or outside a closed curve as follows: Draw a ray
1149 from the test point in any direction and count the number of path segments that cross the ray,
1150 regardless of the direction. If the number is odd, the point is inside; if the number is even, the point is
1151 outside.



1152
1153 **RULE_WINDING:** Specifies that areas are filled according to the nonzero winding rule. According to this
1154 rule, it can be determined whether a test point is inside or outside a closed curve as follows: draw a ray
1155 from that point to infinity in any direction and examine the places where a segment of the path crosses
1156 the ray. Starting with a count of 0, the rule adds 1 each time a curve segment crosses the ray from left
1157 to right and subtracts 1 each time a segment crosses from right to left. After counting all the crossings,
1158 if the result is 0, the point is outside the path; otherwise, it is inside.



1159
1160

1161 **Sub-elements:** N/A

1162 **Note:**

1163 4.11.2.11 RENDER_MODE

1164 **Semantics:** Set the current render mode (line, fill, clip, or their combination)

1165 **Properties:**

1166 **v1:** a character string, representing the render mode. The possible values for this property are LINE,
1167 FILL, CLIP, or some combination of the three, with values separated by a comma.

1168 **LINE:** draw a line along the path.

1169 **FILL:** draw the entire region enclosed by the path.

1170 **CLIP:** current clip area will be set as the intersection of the next path graphics and current clip area.

1171 **Sub-elements:** N/A

1172 4.11.2.12 RASTER_OP

1173 **Semantics:** Set the current raster operation.

1174 **Properties:**

1175 **v1:** a character string, representing the raster operation. The possible values for this property are
1176 ROP_COPY, ROP_N_COPY, ROP_RESET, ROP_SET, ROP_NOP, ROP_REV, ROP_AND, ROP_AND_N,
1177 ROP_N_AND, ROP_N_AND_N, ROP_OR, ROP_OR_N, ROP_N_OR, ROP_N_OR_N, ROP_XOR, and
1178 ROP_EOR. In the following, 'pixel_color' represents the color after a raster operation; 'src' is the
1179 currently used color; 'dest' is the current color of the destination bitmap to be drawn upon; '&' is
1180 bitwise AND; '|' is bitwise OR; '^' is bitwise XOR; and '~' is bitwise NOT, which has the highest priority
1181 over the other logical operators.

1182

1183 **ROP_COPY:** pixel_color = src

1184 **ROP_N_COPY:** pixel_color = ~src

1185 **ROP_RESET:** pixel_color = 0 (all bits of pixel_color are set zero)

1186 **ROP_SET:** pixel_color = 1 (all bits of pixel_color are set 1)

1187 **ROP_NOP:** pixel_color = dest

1188 **ROP_REV:** pixel_color = ~dest

1189 ROP_AND: pixel_color = src & dest
1190 ROP_AND_N: pixel_color = src & ~dest
1191 ROP_N_AND: pixel_color = ~src & dest
1192 ROP_N_AND_N: pixel_color = ~src & ~dest
1193 ROP_OR: pixel_color = src | dest
1194 ROP_OR_N: pixel_color = src | ~dest
1195 ROP_N_OR: pixel_color = ~src | dest
1196 ROP_N_OR_N: pixel_color = ~src | ~dest
1197 ROP_XOR: pixel_color = src ^ dest
1198 ROP_EOR: pixel_color = src ^ ~dest

1199 **Sub-elements:** N/A

1200 4.11.2.13 TEXT_DIR

1201 **Semantics:** Set the current text direction. The direction specifies that line along which successive character
1202 origin points are placed (see figure 4); that is the line from one glyph origin to the next glyph origin.

1203 **Properties:**

1204 v1: a character string, representing the text direction. The possible values for this property are
1205 HEAD_LEFT, HEAD_RIGHT, HEAD_TOP and HEAD_BOTTOM. HEAD_LEFT is the text direction is from left
1206 to right. HEAD_RIGHT is the text direction is from right to left. HEAD_TOP is the text direction is from
1207 top to bottom. HEAD_BOTTOM is the text direction is from bottom to top.

1208 **Sub-elements:** N/A

1209 4.11.2.14 CHAR_DIR

1210 **Semantics:** Set the current character direction (e.g., the direction in which a character is rendered). The
1211 heading direction is from the bottom of a character to the top.

1212 **Properties:**

1213 v1: a character string representing the character direction. The possible values for this property are
1214 HEAD_LEFT, HEAD_RIGHT, HEAD_TOP and HEAD_BOTTOM. HEAD_LEFT is the character's heading
1215 direction is left. HEAD_RIGHT is the character's heading direction is right. HEAD_TOP is the character's
1216 heading direction is up. HEAD_BOTTOM is the character's heading direction is down.

1217 **Sub-elements:** N/A

1218 4.11.2.15 CHAR_ROTATE

1219 **Semantics:** Set the current character rotation angle.

1220 **Properties:**

1221 v1: a floating point number, representing the character rotating radian. A positive value represents
1222 counterclockwise; a negative value represents clockwise.

1223 v2: a character string, representing whether the rotation is around the character center or around the
1224 top-left corner. The possible values for this property are ROT_CENTER and ROT_LEFTTOP.

1225 **Sub-elements:** N/A

1226 4.11.2.16 CHAR_SLANT

1227 **Semantics:** Set the slant of the character.

1228 **Properties:**

1229 $v1$: a floating point number, representing the character slanting radian, regardless of reading direction.
1230 $0 \sim \pi/2$ represents right slant, $3\pi/2 \sim 2\pi$ represents left slant, and 0 represents non-slant; other values
1231 are not used.

1232 **Sub-elements:** N/A

1233 4.11.2.17 CHAR_SIZE

1234 **Semantics:** Set the current character width and height.

1235 **Properties:**

1236 $v1$: a positive floating point number, representing the character width.

1237 $v2$: a positive floating point number, representing the character height.

1238 **Sub-elements:** N/A

1239 4.11.2.18 CHAR_WEIGHT

1240 **Semantics:** Set the current character weight. The default value is 0. The thickness of a character stroke shall be
1241 the normal thickness plus $\text{weight} * (\text{character height})$. The minimum thickness of a character's stroke is zero.

1242 **Properties:**

1243 $v1$: a floating point number, ranging between -1 to 1, inclusively, representing the character weight.

1244 **Sub-elements:** N/A

1245 4.11.2.19 CHAR_STYLE

1246 **Semantics:** Set the current character style.

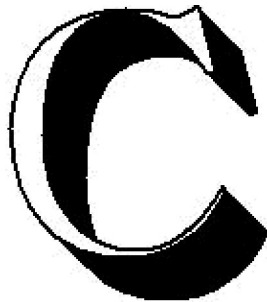
1247 **Properties:**

1248 $v1$: a character string, representing the character style. The possible values for this property are
1249 SHADOW, HOLLOW and OUTLINE, or some combination of the three, separated by commas. If the
1250 string is set to empty, then any previous setting is cleared.

1252 SHADOW: set shadow style. If this character style is set, then the following algorithm is used to render
1253 the shadow effect:

- 1255 • If SHADOW_NEG (§4.11.2.30) is false, the character is extended with a distance of
1256 SHADOW_LEN (§4.11.2.27) along the shadow direction (§4.11.2.28), then a hollowed character
1257 with raster operation ROP_COPY is drawn in the original position. The border width of the
1258 hollowed character is SHADOW_WIDTH (§4.11.2.26).
- 1259 • If SHADOW_NEG is true, the character position is moved with a distance of SHADOW_LEN
1260 along the shadow direction, and extended SHADOW_WIDTH along the shadow direction; then
1261 the character is drawn in the original position with background color and raster operation
1262 ROP_COPY, and extended with a distance SHADOW_LEN along the shadow direction; then in
1263 the original position, a character with normal color and raster operation ROP_COPY is drawn.
1264

1265

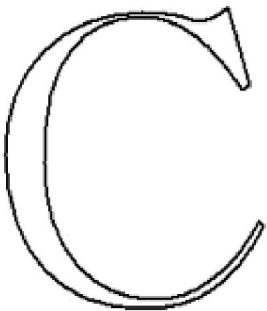


1266

1267

1268

HOLLOW: set hollow style. If this character style is set, a line with thickness HOLLOW_BORDER (§4.11.2.35) should be drawn along the outline of the character.



1269

1270

1271

1272

OUTLINE: set outline style. If this character style is set, a line with thickness OUTLINE_BORDER (§4.11.2.33), and with distance OUTLINE_WIDTH (§4.11.2.34) from the outline of the character, should be drawn along the outline of the character.

1273

1274

Sub-elements: N/A

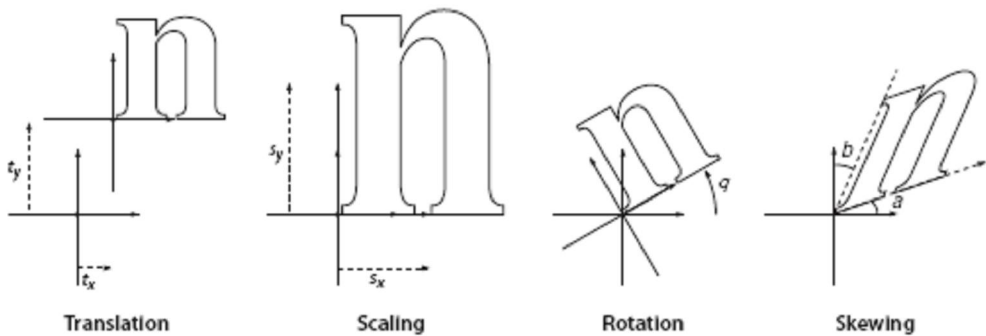
1275

1276 **4.11.2.20 TEXT_MATRIX**

1277

1278

Semantics: Set the current text transformation matrix. This command applies to each character individually within a TEXT object. The visual effect of transforming a character is shown below:



1279

1280

Properties: N/A

1281

Sub-elements:

1282

matrix: element of the MATRIX (§4.11.3.2) type, responsible for transforming coordinates of text.

1283 **4.11.2.21 IMAGE_MATRIX**

1284 **Semantics:** Set the current image transformation matrix

1285 **Properties:** N/A

1286 **Sub-elements:**

1287 *matrix*: element of MATRIX (§4.11.3.2) type, used for transforming coordinates of an image.

1288 **4.11.2.22 GRAPH_MATRIX**

1289 **Semantics:** Set the current line/curve transformation matrix

1290 **Properties:** N/A

1291 **Sub-elements:**

1292 *matrix*: element of the MATRIX (§4.11.3.2) type, used for transforming the coordinates of path
1293 graphics, such as line, Bezier curve, arc, circle, ellipse, rect, roundrect, subpath, path, etc.

1294 **4.11.2.23 EXT_MATRIX**

1295 **Semantics:** Set the current extension transformation matrix

1296 **Properties:** N/A

1297 **Sub-elements:**

1298 *matrix*: element of the MATRIX (§4.11.3.2) type, used for transforming the coordinates of all path
1299 graphics, images and texts. The current extension transformation matrix is applied to the object after
1300 any current dedicated transformation matrix has been applied to the object.

1301 **4.11.2.24 PUSH_GS**

1302 **Semantics:** Push the current graphics state onto the graphics state stack.

1303 **Properties:** N/A

1304 **Sub-elements:** N/A

1305 **4.11.2.25 POP_GS**

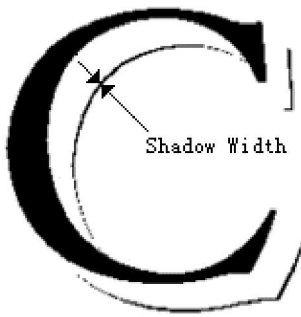
1306 **Semantics:** Pop out the top value from the graphics state stack, replacing the current graphics state.

1307 **Properties:** N/A

1308 **Sub-elements:** N/A

1309 **4.11.2.26 SHADOW_WIDTH**

1310 **Semantics:** Set the border width of the current character shadow. SHADOW_WIDTH represents the thickness of
1311 the outline of a shadow.



1312

1313 **Properties:**

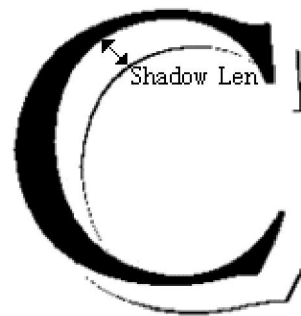
1314 *v1*: a non-negative floating point number, representing the shadow border width.

1315 **Sub-elements:** N/A

1316

1317 4.11.2.27 SHADOW_LEN

1318 **Semantics:** Set the length of the current character shadow. SHADOW_LEN represents the displacement of the
 1319 shadow with respect to the character.



1320

1321 **Properties:**

1322 *v1*: a non-negative floating point number, representing the character shadow length.

1323 **Sub-elements:** N/A

1324

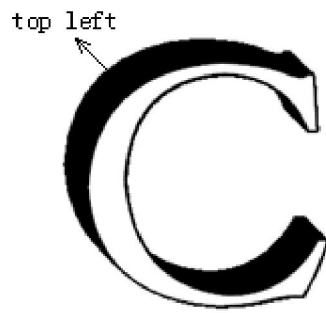
1325 4.11.2.28 SHADOW_DIR

1326 **Semantics:** Set the direction of the current character shadow

1327 **Properties:**

1328 *v1*: a character string. The possible values for this property are SHADOW_LT, SHADOW_LB,
 1329 SHADOW_RT and SHADOW_RB. Choosing one of these values specifies which direction the character
 1330 shadow will be seen.

1331 SHADOW_LT: the character shadow direction is top left.

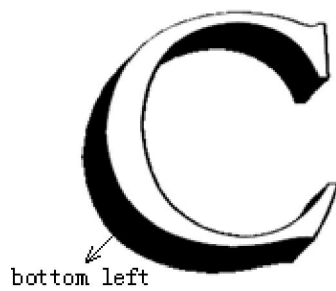


1332

1333

1334

SHADOW_LB: the character shadow direction is bottom left.



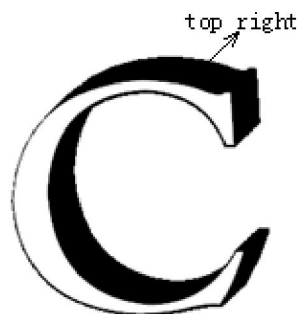
1335

1336

1337

1338

SHADOW_RT: the character shadow direction is top right.

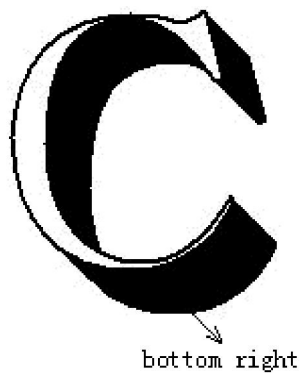


1339

1340

1341

SHADOW_RB: the character shadow direction is bottom right.



1342

1343

1344 **Sub-elements:** N/A

1345

1346 4.11.2.29 SHADOW_ATL

1347 **Semantics:** Set whether to adjust the coordinates of a character when the direction of character shadow is to
1348 the left or bottom.

1349 **Properties:**

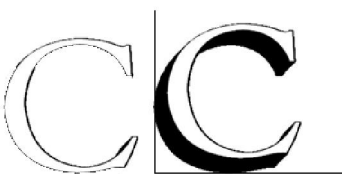
1350 **v1:** a Boolean value, representing whether to alter the coordinates of a character. The value 'true'
1351 specifies that the coordinates are altered.

1352 **Sub-elements:** N/A

1353 [Example: Illustrated in the figures below, when a character is shadowed, the bounding box of its outline is
1354 bigger. If two characters that are not shadowed are adjacent, their baselines are aligned horizontally. A shadow
1355 effect will break this horizontal alignment. Also, a shadow to the left will occupy the space between this
1356 character and its left neighbor. When a rendering engine draws the character, it can position the character
1357 based on the specific coordinate; or it can adjust the coordinate so that the bottom left point of the shadowed
1358 character's outline bounding box moves to the specific coordinate. This is made by offset x or y coordinates by
1359 the distance of SHADOW_LEN divided by the square root of 2. When the shadow is to the bottom of the
1360 character, subtract y by the distance; when the shadow is to the left, add x by the distance. Make both
1361 adjustments when the shadow is to the bottom left. This explains the parameter SHADOW_ATL. When
1362 SHADOW_ATL is false, the specific coordinate is used without adjustment; when it is true, an adjustment
1363 should be made. The first figure illustrates the effect before adjustment, while the second figure illustrates the
1364 effect after adjustment.



1365



1366

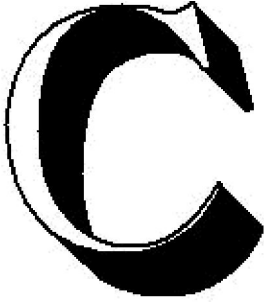
1367

1368

1369 *end example]*

1370 4.11.2.30 SHADOW_NEG

1371 **Semantics:** Set the current shadow character as an intaglio character as illustrated in the following figures.



1372

1373 SHADOW_NEG is false



1374

1375 SHADOW_NEG is true

1376

1377 **Properties:**

1378 v1: a boolean value, representing whether the current shadow character is an intaglio character. A
1379 'true' value specifies an intaglio character.

1380 **Sub-elements:** N/A

1381 4.11.2.31 CLIP_AREA

1382 **Semantics:** Set the current clip area

1383 **Properties:** N/A

1384 **Sub-elements:**

1385 cliparea: PATH type, representing the new clip area.

1386 The Path specified by a CLIP_AREA command object is relative to the page. The portions of graphic
1387 objects that lie outside of the current clip area are not rendered.

1388 4.11.2.32 FONT

1389 **Semantics:** set the font used by an encoding/character set. [*Example:* set an English character to use the font
1390 named "Arial". *end example*]

1391

1392 **Properties:**

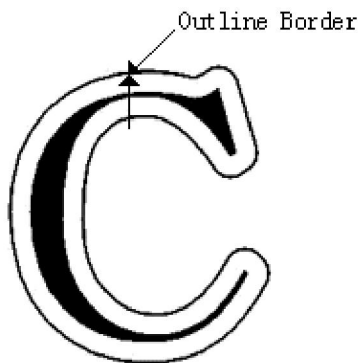
1393 v1: a character string, representing the encoding/character set. The valid value for this property is the
1394 same as for the *encode* property of TEXT (§4.10.11).

1395 v2: a character string, representing the font that will be used by the encoding/character set.

1396 **Sub-elements:** N/A

1397 4.11.2.33 **OUTLINE_BORDER**

1398 **Semantics:** Set the border width of the current outline character



1399
1400 **Properties:**
1401 *v1*: a non-negative floating point number, representing the border width.

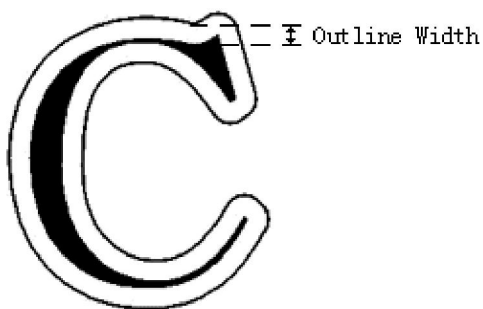
1402 **Sub-elements:** N/A

1403

1404

1405 4.11.2.34 **OUTLINE_WIDTH**

1406 **Semantics:** Set the outline width of the current outline character



1407
1408 **Properties:**
1409 *v1*: a non-negative floating point number, representing the outline width.

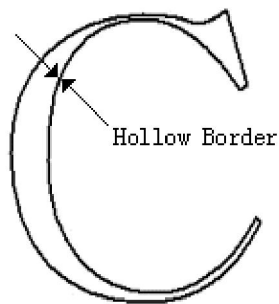
1410 **Sub-elements:** N/A

1411

1412

1413 4.11.2.35 **HOLLOW_BORDER**

1414 **Semantics:** Set the border width of the current hollow character



1415

1416 **Properties:**

1417 *v1*: a non-negative floating point number, representing the border width.

1418 **Sub-elements:** N/A

1419

1420 4.11.3 Definition of Referenced Type

1421 This clause specifies the definition of the data types referred in the UOML XML schema descriptions.

1422 4.11.3.1 COLOR_RGB

1423 **Semantics:** the value of a color setting

1424 **Properties:**

1425 *r*: red component

1426 *g*: green component

1427 *b*: blue component

1428 *a*: optional alpha component.

1429 **Sub-element:** N/A

1430 4.11.3.2 MATRIX

1431 **Semantics:** the values in a transformation matrix

1432 **Properties:**

1433 *f11*: floating point number

1434 *f12*: floating point number

1435 *f21*: floating point number

1436 *f22*: floating point number

1437 *f31*: floating point number

1438 *f32*: floating point number

1439 **Sub-element:** N/A

1440 **[Note:**

1441 A transformation of matrix in UOML is specified by six numbers. In an abbreviated notation, this array
 1442 is denoted [*f11 f12 f21 f22 f31 f32*]; it can represent any linear transformation from one coordinate
 1443 system to another. The transformation is carried out as follows:

1444

1445 $x' = f11 \times x + f21 \times y + f31$
 1446 $y' = f12 \times x + f22 \times y + f32$

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

1458

end note]

1459

4.12 Default Value of Graphics State

State	Default Value
line color	Black
fill color	Black
character shadow color	Black
character outline color	Black
text color	Black
line width	1
line cap style	END_BUT
line join style	JOIN_MITER
miter limit	10
fill rule	RULE_WINDING
render mode	LINE
raster operation	ROP_COPY
text direction	HEAD_LEFT
character direction	HEAD_TOP
character rotation	ROT_CENTER, no rotation
character slant	Non-slant
character width	Undefined
character height	Undefined
character weight	0
character style	Normal style (no shadow, not hollow, no outline)
text transformation matrix	Identity matrix ([1,0,0,1,0,0])
image transformation matrix	Identity matrix

path graphics transformation matrix	Identity matrix
extension transformation matrix	Identity matrix
clip area	Current page
font	Undefined

1460
1461

1462 4.13 Definition of Parameter Data Types

1463 This clause specifies the definition of the data types referenced in the UOML XML schema definition.

1464 4.13.1 INT

1465 Properties:

1466 *name*: a character string value, xs:string type

1467 *val*: xs:integer type

1468 **Sub-element:** N/A

1469 4.13.2 DOUBLE

1470 Properties:

1471 *name*: a character string, xs:string type

1472 *val*: xs:double type

1473 **Sub-element:** N/A

1474 4.13.3 LONG

1475 Properties:

1476 *name*: a character string, xs:string type

1477 *val*: xs:long type

1478 **Sub-element:** N/A

1479 4.13.4 DATE

1480 Properties:

1481 *name*: a character string, xs:string type

1482 *val*: xs:date type

1483 **Sub-element:** N/A

1484 4.13.5 TIME

1485 Properties:

1486 *name*: a character string, xs:string type

1487 *val*: xs:time type

1488 **Sub-element:** N/A

1489 4.13.6 DATETIME

1490 Properties:

1491 *name*: a character string, xs:string type

1492 *val*: xs:datetime type

1493 Sub-element: N/A

1494 4.13.7 DURATION

1495 Properties:

1496 *name*: a character string, xs:string type

1497 *val*: xs:duration type

1498 Sub-element: N/A

1499 4.13.8 STRING

1500 Properties:

1501 *name*: a character string, xs:string type

1502 *val*: xs:string type

1503 Sub-element: N/A

1504 4.13.9 BINARY

1505 Properties:

1506 *name*: a character string, xs:string type

1507 *val*: xs:base64Binary type

1508 Sub-element: N/A

1509 4.13.10 BOOL

1510 Properties:

1511 *name*: a character string, xs:string type

1512 *val*: xs:boolean type

1513 Sub-element: N/A

1514 4.13.11 COMPOUND

1515 Property:

1516 *name*: a character string, xs:string type

1517 Sub-element:

1518 *arc*: ARC type

1519 *bezier*: BEZIER type

1520 *circle*: CIRCLE type

1521 *cmd*: CMD type

1522 *rgb*: COLOR_RGB type
1523 *doc*: DOC type
1524 *docbase*: DOCTYPE type
1525 *docset*: DOCSET type
1526 *ellipse*: ELLIPSE type
1527 *embedfont*: EMBEDFONT type
1528 *fontlist*: FONTLIST type
1529 *fontmap*: FONTMAP type
1530 *image*: IMAGE type
1531 *layer*: LAYER type
1532 *line*: LINE type
1533 *matrix*: MATRIX type
1534 *meta*: META type
1535 *metalist*: METALIST type
1536 *page*: PAGE type
1537 *path*: PATH type
1538 *rect*: RECT type
1539 *roundrect*: ROUNDRECT type
1540 *subpath*: SUBPATH type
1541 *text*: TEXT type
1542 *objstream*: OBJSTREAM type

1543 [Note: Each sub-element may occur zero or more times. *end note*]

1544

1545 4.14 Data Ranges

1546 The following are the general rules for data ranges:

1547

- 1548 1. Unless otherwise specified, all numeric values may be positive, negative or zero.
- 1549 2. Positive, negative, or zero values are allowed for coordinates and points in the logical coordinate
1550 system (e.g. -1, 3).
- 1551 3. Integer values are 32-bit precision; the range of integer values is as defined by xs:integer in XML
1552 Schema 1.0 Part 2.
- 1553 4. Float values use double-precision; the valid range is as defined by xs:double in XML Schema 1.0
1554 Part 2.
- 1555 5. API calls that set values outside a valid range (either specifically specified or within the ranges
1556 above) will fail with a return of RET.

- 1557 6. A special case is COLOR_RGB. RGB32 is used, thus each property of COLOR_RGB(r, g, b, a) falls
1558 within a range of 0-255.
1559 7. Valid ranges and formats for a date are as defined by xs:date in XML Schema 1.0 Part 2.
1560

1561

5. Conformance

1562

1563

1564

1565

The text in this OASIS standard is divided into *normative* and *informative* categories. Unless documented otherwise, all features specified in normative text of this OASIS standard shall be implemented. Text marked informative (using the mechanisms described in §1.5) is for information purposes only. Unless stated otherwise, all text is normative.

1566

Use of the word “shall” indicates required behavior.

1567

Any behavior that is not explicitly specified by this OASIS standard is implicitly unspecified (§4).

1568

5.1.1 DCMS Conformance

1569

1570

1571

A UOML Document Management System (DCMS) has conformance if it implements all of the UOML instructions in compliance with the syntax as described in the schema [UOMLSchema] and semantics in this OASIS standard.

1572

5.1.2 Application Conformance

1573

A UOML application is conformant if both of the following are true:

1574

1575

1576

- The application issues UOML instructions as schema-valid XML] as specified in this OASIS standard to the DCMS; and
- The application parses the return instructions from the DCMS according to this OASIS standard.

1577

Annex A.UOML XML Schema

1578

This annex is informative.

1579

The following is a copy of the XML Schema for UOML for ancillary purposes. It describes the types and elements, in XML format, for UOML. The normative schema is provided with the specification.

1580

1581

The normative XML schema definition is located at: <http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-part1-v1.0-schema-errata.xsd>.

1582

1583

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

1584

```
<xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

1585

```
xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0"
```

1586

```
targetNamespace="urn:oasis:names:tc:uoml:xmlns:uoml:1.0"
```

1587

```
elementFormDefault="unqualified" attributeFormDefault="unqualified">
```

1588

```
  <xs:complexType name="ARC">
```

1589

```
    <xs:annotation>
```

1590

```
      <xs:documentation>arc</xs:documentation>
```

1591

```
    </xs:annotation>
```

1592

```
    <xs:attribute name="clockwise" type="xs:boolean" use="required"/>
```

1593

```
    <xs:attribute name="start" type="xs:string" use="required"/>
```

1594

```
    <xs:attribute name="end" type="xs:string" use="required"/>
```

1595

```
    <xs:attribute name="center" type="xs:string" use="required"/>
```

1596

```
    <xs:attribute name="angle" type="xs:float" use="required"/>
```

1597

```
  </xs:complexType>
```

1598

```
  <xs:complexType name="BEZIER">
```

1599

```
    <xs:annotation>
```

1600

```
      <xs:documentation>bezier curve</xs:documentation>
```

1601

```
    </xs:annotation>
```

1602

```
    <xs:attribute name="start" type="xs:string" use="required"/>
```

1603

```
    <xs:attribute name="ctrl" type="xs:string" use="required"/>
```

1604

```
    <xs:attribute name="ctrl2" type="xs:string" use="optional"/>
```

1605

```
    <xs:attribute name="end" type="xs:string" use="required"/>
```

1606

```
  </xs:complexType>
```

1607

```
  <xs:complexType name="CIRCLE">
```

1608

```
    <xs:annotation>
```

1609

```
      <xs:documentation>circle</xs:documentation>
```

1610

```
    </xs:annotation>
```

1611

```
    <xs:attribute name="radius" type="xs:int" use="required"/>
```

1612

```
    <xs:attribute name="center" type="xs:string" use="required"/>
```

1613

```
  </xs:complexType>
```

1614

```
  <xs:complexType name="LINE">
```

1615

```
    <xs:annotation>
```

1616

```
      <xs:documentation>line</xs:documentation>
```

1617

```
    </xs:annotation>
```

```

1618         <xs:attribute name="start" type="xs:string" use="required"/>
1619         <xs:attribute name="end" type="xs:string" use="required"/>
1620     </xs:complexType>
1621     <xs:complexType name="RECT">
1622         <xs:annotation>
1623             <xs:documentation>rect</xs:documentation>
1624         </xs:annotation>
1625         <xs:attribute name="tl" type="xs:string" use="required"/>
1626         <xs:attribute name="br" type="xs:string" use="required"/>
1627     </xs:complexType>
1628     <xs:complexType name="ELLIPSE">
1629         <xs:annotation>
1630             <xs:documentation>ellipse</xs:documentation>
1631         </xs:annotation>
1632         <xs:attribute name="xr" type="xs:int" use="required"/>
1633         <xs:attribute name="yr" type="xs:int" use="required"/>
1634         <xs:attribute name="center" type="xs:string" use="required"/>
1635         <xs:attribute name="angle" type="xs:float" use="required"/>
1636     </xs:complexType>
1637     <xs:complexType name="ROUNDRECT">
1638         <xs:annotation>
1639             <xs:documentation>roundrect</xs:documentation>
1640         </xs:annotation>
1641         <xs:attribute name="xr" type="xs:int" use="required"/>
1642         <xs:attribute name="yr" type="xs:int" use="required"/>
1643         <xs:attribute name="tl" type="xs:string" use="required"/>
1644         <xs:attribute name="br" type="xs:string" use="required"/>
1645     </xs:complexType>
1646     <xs:complexType name="META">
1647         <xs:annotation>
1648             <xs:documentation>metadata</xs:documentation>
1649         </xs:annotation>
1650         <xs:attribute name="key" type="xs:string" use="required"/>
1651         <xs:attribute name="val" type="xs:string" use="required"/>
1652     </xs:complexType>
1653     <xs:complexType name="METALIST">
1654         <xs:annotation>
1655             <xs:documentation>metadata list</xs:documentation>
1656         </xs:annotation>
1657         <xs:sequence>
1658             <xs:element name="meta" type="uoml:META" minOccurs="0"
1659 maxOccurs="unbounded"/>
1660         </xs:sequence>
1661     </xs:complexType>
1662     <xs:complexType name="CMD">
1663         <xs:annotation>
1664             <xs:documentation>cmd</xs:documentation>
1665         </xs:annotation>
1666         <xs:sequence minOccurs="0">

```



```

1667         <xs:choice>
1668             <xs:element name="cliparea" type="uoml:PATH"/>
1669             <xs:element name="matrix" type="uoml:MATRIX"/>
1670             <xs:element name="rgb" type="uoml:COLOR_RGB"/>
1671         </xs:choice>
1672     </xs:sequence>
1673     <xs:attribute name="name" type="uoml:CMDNAME" use="required"/>
1674     <xs:attribute name="v1" type="xs:anySimpleType"/>
1675     <xs:attribute name="v2" type="xs:anySimpleType"/>
1676 </xs:complexType>
1677 <xs:complexType name="MATRIX">
1678     <xs:annotation>
1679         <xs:documentation>matrix</xs:documentation>
1680     </xs:annotation>
1681     <xs:attribute name="f11" type="xs:float" use="required"/>
1682     <xs:attribute name="f12" type="xs:float" use="required"/>
1683     <xs:attribute name="f21" type="xs:float" use="required"/>
1684     <xs:attribute name="f22" type="xs:float" use="required"/>
1685     <xs:attribute name="f31" type="xs:float" use="required"/>
1686     <xs:attribute name="f32" type="xs:float" use="required"/>
1687 </xs:complexType>
1688 <xs:complexType name="SUBPATH">
1689     <xs:annotation>
1690         <xs:documentation>subpath</xs:documentation>
1691     </xs:annotation>
1692     <xs:attribute name="data" type="xs:string" use="required"/>
1693 </xs:complexType>
1694 <xs:complexType name="PATH">
1695     <xs:annotation>
1696         <xs:documentation>path</xs:documentation>
1697     </xs:annotation>
1698     <xs:sequence>
1699         <xs:choice minOccurs="0" maxOccurs="unbounded">
1700             <xs:element name="subpath" type="uoml:SUBPATH"/>
1701             <xs:element name="rect" type="uoml:RECT"/>
1702             <xs:element name="circle" type="uoml:CIRCLE"/>
1703             <xs:element name="ellipse" type="uoml:ELLIPSE"/>
1704             <xs:element name="roundrect" type="uoml:ROUNDRECT"/>
1705         </xs:choice>
1706     </xs:sequence>
1707 </xs:complexType>
1708 <xs:complexType name="COLOR_RGB">
1709     <xs:annotation>
1710         <xs:documentation>rgb color</xs:documentation>
1711     </xs:annotation>
1712     <xs:attribute name="r" type="xs:short" use="required"/>
1713     <xs:attribute name="g" type="xs:short" use="required"/>
1714     <xs:attribute name="b" type="xs:short" use="required"/>
1715     <xs:attribute name="a" type="xs:short" use="optional"/>

```

```

1716 </xs:complexType>
1717 <xs:complexType name="EMBEDFONT">
1718   <xs:annotation>
1719     <xs:documentation>embedded font</xs:documentation>
1720   </xs:annotation>
1721   <xs:simpleContent>
1722     <xs:extension base="xs:base64Binary">
1723       </xs:extension>
1724     </xs:simpleContent>
1725   </xs:complexType>
1726 <xs:complexType name="FONTMAP">
1727   <xs:annotation>
1728     <xs:documentation>font mapping</xs:documentation>
1729   </xs:annotation>
1730   <xs:attribute name="name" type="xs:string" use="required"/>
1731   <xs:attribute name="no" type="xs:int" use="required"/>
1732 </xs:complexType>
1733 <xs:complexType name="FONTLIST">
1734   <xs:annotation>
1735     <xs:documentation>font list</xs:documentation>
1736   </xs:annotation>
1737 </xs:complexType>
1738 <xs:complexType name="IMAGE">
1739   <xs:annotation>
1740     <xs:documentation>image</xs:documentation>
1741   </xs:annotation>
1742   <xs:simpleContent>
1743     <xs:extension base="xs:base64Binary">
1744       <xs:attribute name="tl" type="xs:string" use="required"/>
1745       <xs:attribute name="br" type="xs:string" use="required"/>
1746       <xs:attribute name="type" type="xs:string" use="required"/>
1747       <xs:attribute name="path" type="xs:string" use="optional"/>
1748     </xs:extension>
1749   </xs:simpleContent>
1750 </xs:complexType>
1751 <xs:complexType name="TEXT">
1752   <xs:annotation>
1753     <xs:documentation>text</xs:documentation>
1754   </xs:annotation>
1755   <xs:attribute name="origin" type="xs:string" use="required"/>
1756   <xs:attribute name="encode" type="xs:string" use="required"/>
1757   <xs:attribute name="text" type="xs:string" use="required"/>
1758   <xs:attribute name="spaces" type="xs:string" use="optional"/>
1759 </xs:complexType>
1760 <xs:simpleType name="CMDNAME">
1761   <xs:annotation>
1762     <xs:documentation>command names</xs:documentation>
1763   </xs:annotation>

```

```

1764     <xs:restriction base="xs:string">
1765         <xs:enumeration value="COLOR_LINE"/>
1766         <xs:enumeration value="COLOR_FILL"/>
1767         <xs:enumeration value="COLOR_TEXT"/>
1768         <xs:enumeration value="COLOR_SHADOW"/>
1769         <xs:enumeration value="COLOR_OUTLINE"/>
1770         <xs:enumeration value="LINE_WIDTH"/>
1771         <xs:enumeration value="LINE_JOIN"/>
1772         <xs:enumeration value="LINE_CAP"/>
1773         <xs:enumeration value="MITER_LIMIT"/>
1774         <xs:enumeration value="FILL_RULE"/>
1775         <xs:enumeration value="RENDER_MODE"/>
1776         <xs:enumeration value="RASTER_OP"/>
1777         <xs:enumeration value="TEXT_DIR"/>
1778         <xs:enumeration value="CHAR_DIR"/>
1779         <xs:enumeration value="CHAR_ROTATE"/>
1780         <xs:enumeration value="CHAR_SLANT"/>
1781         <xs:enumeration value="CHAR_SIZE"/>
1782         <xs:enumeration value="CHAR_WEIGHT"/>
1783         <xs:enumeration value="CHAR_STYLE"/>
1784         <xs:enumeration value="TEXT_MATRIX"/>
1785         <xs:enumeration value="IMAGE_MATRIX"/>
1786         <xs:enumeration value="GRAPH_MATRIX"/>
1787         <xs:enumeration value="EXT_MATRIX"/>
1788         <xs:enumeration value="PUSH_GS"/>
1789         <xs:enumeration value="POP_GS"/>
1790         <xs:enumeration value="SHADOW_WIDTH"/>
1791         <xs:enumeration value="SHADOW_DIR"/>
1792         <xs:enumeration value="SHADOW_LEN"/>
1793         <xs:enumeration value="SHADOW_NEG"/>
1794         <xs:enumeration value="SHADOW_ATL"/>
1795         <xs:enumeration value="CLIP_AREA"/>
1796         <xs:enumeration value="FONT"/>
1797         <xs:enumeration value="OUTLINE_BORDER"/>
1798         <xs:enumeration value="OUTLINE_WIDTH"/>
1799         <xs:enumeration value="HOLLOW_BORDER"/>
1800     </xs:restriction>
1801 </xs:simpleType>
1802 <xs:simpleType name="LINECAP">
1803     <xs:annotation>
1804         <xs:documentation>line cap style</xs:documentation>
1805     </xs:annotation>
1806     <xs:restriction base="xs:string">
1807         <xs:enumeration value="END_BUTT"/>
1808         <xs:enumeration value="END_SQUARE"/>
1809         <xs:enumeration value="END_ROUND"/>
1810     </xs:restriction>
1811 </xs:simpleType>
1812 <xs:simpleType name="JOINCAP">

```

```

1813         <xs:annotation>
1814             <xs:documentation>line join style</xs:documentation>
1815         </xs:annotation>
1816         <xs:restriction base="xs:string">
1817             <xs:enumeration value="JOIN_MITER"/>
1818             <xs:enumeration value="JOIN_BEVEL"/>
1819             <xs:enumeration value="JOIN_ROUND"/>
1820         </xs:restriction>
1821     </xs:simpleType>
1822     <xs:simpleType name="FILLRULE">
1823         <xs:annotation>
1824             <xs:documentation>fill rule</xs:documentation>
1825         </xs:annotation>
1826         <xs:restriction base="xs:string">
1827             <xs:enumeration value="RULE_EVENODD"/>
1828             <xs:enumeration value="RULE_WINDING"/>
1829         </xs:restriction>
1830     </xs:simpleType>
1831     <xs:simpleType name="ROP">
1832         <xs:annotation>
1833             <xs:documentation>rop operation</xs:documentation>
1834         </xs:annotation>
1835         <xs:restriction base="xs:string">
1836             <xs:enumeration value="ROP_COPY"/>
1837             <xs:enumeration value="ROP_N_COPY"/>
1838             <xs:enumeration value="ROP_RESET"/>
1839             <xs:enumeration value="ROP_SET"/>
1840             <xs:enumeration value="ROP_NOP"/>
1841             <xs:enumeration value="ROP_REV"/>
1842             <xs:enumeration value="ROP_AND"/>
1843             <xs:enumeration value="ROP_AND_N"/>
1844             <xs:enumeration value="ROP_N_AND"/>
1845             <xs:enumeration value="ROP_N_AND_N"/>
1846             <xs:enumeration value="ROP_OR"/>
1847             <xs:enumeration value="ROP_OR_N"/>
1848             <xs:enumeration value="ROP_N_OR"/>
1849             <xs:enumeration value="ROP_N_OR_N"/>
1850             <xs:enumeration value="ROP_XOR"/>
1851             <xs:enumeration value="ROP_EOR"/>
1852         </xs:restriction>
1853     </xs:simpleType>
1854     <xs:simpleType name="CHARTXTDIR">
1855         <xs:annotation>
1856             <xs:documentation>text or char direction</xs:documentation>
1857         </xs:annotation>
1858         <xs:restriction base="xs:string">
1859             <xs:enumeration value="HEAD_LEFT"/>
1860             <xs:enumeration value="HEAD_RIGHT"/>
1861             <xs:enumeration value="HEAD_TOP"/>

```

```

1862         <xs:enumeration value="HEAD_BOTTOM"/>
1863     </xs:restriction>
1864 </xs:simpleType>
1865 <xs:simpleType name="SHADOWDIR">
1866     <xs:annotation>
1867         <xs:documentation>shadow direction</xs:documentation>
1868     </xs:annotation>
1869     <xs:restriction base="xs:string">
1870         <xs:enumeration value="SHADOW_LT"/>
1871         <xs:enumeration value="SHADOW_LB"/>
1872         <xs:enumeration value="SHADOW_RT"/>
1873         <xs:enumeration value="SHADOW_RB"/>
1874     </xs:restriction>
1875 </xs:simpleType>
1876 <xs:complexType name="OBJSTREAM">
1877     <xs:annotation>
1878         <xs:documentation>object stream</xs:documentation>
1879     </xs:annotation>
1880 </xs:complexType>
1881 <xs:complexType name="LAYER">
1882     <xs:annotation>
1883         <xs:documentation>layer</xs:documentation>
1884     </xs:annotation>
1885 </xs:complexType>
1886 <xs:complexType name="PAGE">
1887     <xs:annotation>
1888         <xs:documentation>page</xs:documentation>
1889     </xs:annotation>
1890     <xs:attribute name="width" type="xs:float" use="required"/>
1891     <xs:attribute name="height" type="xs:float" use="required"/>
1892     <xs:attribute name="resolution" type="xs:int" use="required"/>
1893 </xs:complexType>
1894 <xs:complexType name="DOC">
1895     <xs:annotation>
1896         <xs:documentation>doc</xs:documentation>
1897     </xs:annotation>
1898     <xs:sequence>
1899         <xs:element name="metainfo" type="uoml:METALIST"/>
1900     </xs:sequence>
1901     <xs:attribute name="name" type="xs:string" use="required"/>
1902 </xs:complexType>
1903 <xs:complexType name="DOCSET">
1904     <xs:annotation>
1905         <xs:documentation>doc set</xs:documentation>
1906     </xs:annotation>
1907     <xs:attribute name="name" type="xs:string" use="required"/>
1908 </xs:complexType>
1909 <xs:complexType name="DOCBASE">
1910     <xs:annotation>

```

```

1911         <xs:documentation>doc base</xs:documentation>
1912     </xs:annotation>
1913     <xs:attribute name="name" type="xs:string" use="required"/>
1914     <xs:attribute name="path" type="xs:string" use="required"/>
1915 </xs:complexType>
1916 <xs:element name="CLOSE">
1917     <xs:complexType>
1918         <xs:attribute name="handle" type="xs:string" use="optional"/>
1919     </xs:complexType>
1920 </xs:element>
1921 <xs:element name="DELETE">
1922     <xs:complexType>
1923         <xs:attribute name="handle" type="xs:string" use="optional"/>
1924     </xs:complexType>
1925 </xs:element>
1926 <xs:element name="INSERT">
1927     <xs:complexType>
1928         <xs:choice>
1929             <xs:element name="xobj" type="uoml:COMPOUND"/>
1930         </xs:choice>
1931         <xs:attribute name="handle" type="xs:string"/>
1932         <xs:attribute name="pos" type="xs:int"/>
1933     </xs:complexType>
1934 </xs:element>
1935 <xs:element name="GET">
1936     <xs:complexType>
1937         <xs:choice>
1938             <xs:element name="disp_conf">
1939                 <xs:complexType>
1940                     <xs:sequence>
1941                         <xs:element name="clip" type="uoml:PATH"
1942 minOccurs="0"/>
1943                     </xs:sequence>
1944                     <xs:attribute name="end_layer" type="xs:int"/>
1945                     <xs:attribute name="resolution"
1946 type="xs:int"/>
1947                     <xs:attribute name="format" type="xs:string"/>
1948                     <xs:attribute name="output" type="xs:string"
1949 use="required"/>
1950                     <xs:attribute name="addr" type="xs:string"
1951 use="required"/>
1952                 </xs:complexType>
1953             </xs:element>
1954             <xs:element name="pos">
1955                 <xs:complexType>
1956                     <xs:attribute name="val" type="xs:int"
1957 use="required"/>
1958                 </xs:complexType>
1959             </xs:element>

```

```

1960         <xs:element name="property">
1961             <xs:complexType>
1962                 <xs:attribute name="name" type="xs:string"
1963 use="required"/>
1964             </xs:complexType>
1965         </xs:element>
1966     </xs:choice>
1967     <xs:attribute name="usage" type="xs:string" use="required"/>
1968     <xs:attribute name="handle" type="xs:string"/>
1969 </xs:complexType>
1970 </xs:element>
1971 <xs:element name="SET">
1972     <xs:complexType>
1973         <xs:choice>
1974             <xs:choice minOccurs="0" maxOccurs="unbounded">
1975                 <xs:element name="intVal" type="uoml:INT"/>
1976                 <xs:element name="floatVal" type="uoml:DOUBLE"/>
1977                 <xs:element name="timeVal" type="uoml:TIME"/>
1978                 <xs:element name="dateVal" type="uoml:DATE"/>
1979                 <xs:element name="dateTimeVal"
1980 type="uoml:DATETIME"/>
1981                 <xs:element name="durationVal"
1982 type="uoml:DURATION"/>
1983                 <xs:element name="stringVal" type="uoml:STRING"/>
1984                 <xs:element name="binaryVal" type="uoml:BINARY"/>
1985                 <xs:element name="compoundVal"
1986 type="uoml:COMPOUND"/>
1987                 <xs:element name="boolVal" type="uoml:BOOL"/>
1988             </xs:choice>
1989         </xs:choice>
1990         <xs:attribute name="handle" type="xs:string"/>
1991     </xs:complexType>
1992 </xs:element>
1993 <xs:element name="USE">
1994     <xs:complexType>
1995         <xs:attribute name="handle" type="xs:string" use="required"/>
1996     </xs:complexType>
1997 </xs:element>
1998 <xs:element name="OPEN">
1999     <xs:complexType>
2000         <xs:attribute name="create" type="xs:boolean" default="true"/>
2001         <xs:attribute name="del_exist" type="xs:boolean"
2002 default="false"/>
2003         <xs:attribute name="path" type="xs:string" use="required"/>
2004     </xs:complexType>
2005 </xs:element>
2006 <xs:element name="SYSTEM">
2007     <xs:complexType>
2008         <xs:choice>

```

```

2009         <xs:element name="flush">
2010             <xs:complexType>
2011                 <xs:attribute name="handle"/>
2012                 <xs:attribute name="path"/>
2013             </xs:complexType>
2014         </xs:element>
2015     </xs:choice>
2016 </xs:complexType>
2017 </xs:element>
2018 <xs:element name="RET">
2019     <xs:complexType>
2020         <xs:choice minOccurs="0" maxOccurs="unbounded">
2021             <xs:element name="intVal" type="uoml:INT"/>
2022             <xs:element name="floatVal" type="uoml:DOUBLE"/>
2023             <xs:element name="timeVal" type="uoml:TIME"/>
2024             <xs:element name="dateVal" type="uoml:DATE"/>
2025             <xs:element name="dateTimeVal" type="uoml:DATETIME"/>
2026             <xs:element name="durationVal" type="uoml:DURATION"/>
2027             <xs:element name="stringVal" type="uoml:STRING"/>
2028             <xs:element name="binaryVal" type="uoml:BINARY"/>
2029             <xs:element name="compoundVal" type="uoml:COMPOUND"/>
2030             <xs:element name="boolVal" type="uoml:BOOL"/>
2031             <xs:element name="longVal" type="uoml:LONG"/>
2032         </xs:choice>
2033     </xs:complexType>
2034 </xs:element>
2035 <xs:complexType name="COMPOUND">
2036     <xs:annotation>
2037         <xs:documentation>compound parameter type</xs:documentation>
2038     </xs:annotation>
2039     <xs:choice minOccurs="0">
2040         <xs:element name="arc" type="uoml:ARC"/>
2041         <xs:element name="bezier" type="uoml:BEZIER"/>
2042         <xs:element name="circle" type="uoml:CIRCLE"/>
2043         <xs:element name="cmd" type="uoml:CMD"/>
2044         <xs:element name="rgb" type="uoml:COLOR_RGB"/>
2045         <xs:element name="doc" type="uoml:DOC"/>
2046         <xs:element name="docbase" type="uoml:DOCBASE"/>
2047         <xs:element name="docset" type="uoml:DOCSET"/>
2048         <xs:element name="ellipse" type="uoml:ELLIPSE"/>
2049         <xs:element name="embedfont" type="uoml:EMBEDFONT"/>
2050         <xs:element name="fontlist" type="uoml:FONTLIST"/>
2051         <xs:element name="fontmap" type="uoml:FONTMAP"/>
2052         <xs:element name="image" type="uoml:IMAGE"/>
2053         <xs:element name="layer" type="uoml:LAYER"/>
2054         <xs:element name="line" type="uoml:LINE"/>
2055         <xs:element name="matrix" type="uoml:MATRIX"/>
2056         <xs:element name="meta" type="uoml:META"/>
2057         <xs:element name="metalist" type="uoml:METALIST"/>

```



```

2058         <xs:element name="page" type="uoml:PAGE"/>
2059         <xs:element name="path" type="uoml:PATH"/>
2060         <xs:element name="rect" type="uoml:RECT"/>
2061         <xs:element name="roundrect" type="uoml:ROUNDRECT"/>
2062         <xs:element name="subpath" type="uoml:SUBPATH"/>
2063         <xs:element name="text" type="uoml:TEXT"/>
2064         <xs:element name="objstream" type="uoml:OBJSTREAM"/>
2065     </xs:choice>
2066     <xs:attribute name="name" type="xs:string"/>
2067 </xs:complexType>
2068 <xs:complexType name="STRING">
2069     <xs:annotation>
2070         <xs:documentation>string parameter type</xs:documentation>
2071     </xs:annotation>
2072     <xs:attribute name="val" type="xs:string" use="required"/>
2073     <xs:attribute name="name" type="xs:string"/>
2074 </xs:complexType>
2075 <xs:complexType name="DOUBLE">
2076     <xs:annotation>
2077         <xs:documentation>double precision float parameter
2078 type</xs:documentation>
2079     </xs:annotation>
2080     <xs:attribute name="val" type="xs:double" use="required"/>
2081     <xs:attribute name="name" type="xs:string"/>
2082 </xs:complexType>
2083 <xs:complexType name="DATE">
2084     <xs:annotation>
2085         <xs:documentation>date parameter type</xs:documentation>
2086     </xs:annotation>
2087     <xs:attribute name="val" type="xs:date" use="required"/>
2088     <xs:attribute name="name" type="xs:string"/>
2089 </xs:complexType>
2090 <xs:complexType name="DATETIME">
2091     <xs:annotation>
2092         <xs:documentation>date and time parameter
2093 type</xs:documentation>
2094     </xs:annotation>
2095     <xs:attribute name="val" type="xs:dateTime" use="required"/>
2096     <xs:attribute name="name" type="xs:string"/>
2097 </xs:complexType>
2098 <xs:complexType name="TIME">
2099     <xs:annotation>
2100         <xs:documentation>time parameter type</xs:documentation>
2101     </xs:annotation>
2102     <xs:attribute name="val" type="xs:time" use="required"/>
2103     <xs:attribute name="name" type="xs:string"/>
2104 </xs:complexType>
2105 <xs:complexType name="DURATION">
2106     <xs:annotation>

```

```

2107         <xs:documentation>duration parameter type</xs:documentation>
2108     </xs:annotation>
2109     <xs:attribute name="val" type="xs:duration" use="required"/>
2110     <xs:attribute name="name" type="xs:string"/>
2111 </xs:complexType>
2112 <xs:complexType name="BINARY">
2113     <xs:annotation>
2114         <xs:documentation>binary parameter type</xs:documentation>
2115     </xs:annotation>
2116     <xs:attribute name="val" type="xs:base64Binary" use="required"/>
2117     <xs:attribute name="name" type="xs:string"/>
2118 </xs:complexType>
2119 <xs:complexType name="INT">
2120     <xs:annotation>
2121         <xs:documentation>integer parameter type</xs:documentation>
2122     </xs:annotation>
2123     <xs:attribute name="val" type="xs:int" use="required"/>
2124     <xs:attribute name="name" type="xs:string"/>
2125 </xs:complexType>
2126 <xs:complexType name="BOOL">
2127     <xs:annotation>
2128         <xs:documentation>boolean parameter type</xs:documentation>
2129     </xs:annotation>
2130     <xs:attribute name="val" type="xs:boolean" use="required"/>
2131     <xs:attribute name="name" type="xs:string"/>
2132 </xs:complexType>
2133 <xs:complexType name="LONG">
2134     <xs:annotation>
2135         <xs:documentation>long parameter type</xs:documentation>
2136     </xs:annotation>
2137     <xs:attribute name="name" type="xs:string"/>
2138     <xs:attribute name="val" type="xs:long" use="required"/>
2139 </xs:complexType>
2140 <xs:simpleType name="CHARSTYLE">
2141     <xs:restriction base="xs:string">
2142         <xs:enumeration value="SHADOW"/>
2143         <xs:enumeration value="HOLLOW"/>
2144         <xs:enumeration value="OUTLINE"/>
2145     </xs:restriction>
2146 </xs:simpleType>
2147 </xs:schema>
2148

```

2149 **End of informative text.**

2150

Annex B. Detailed UOML Examples

2151

This annex is informative.

2152

The examples below demonstrate the usage of many of the UOML instructions. Each example is followed by a corresponding “RET” instruction.

2153

2154

The XML string of a UOML instruction may be preceded by a prolog to specify the character encoding of the XML string. If default encoding is UTF-8, the prolog, `<?xml version="1.0" encoding="UTF-8"?>`, may be omitted. The default namespace for the XML string is: `urn:oasis:names:tc:uoml:xmlns:uoml:1.0`.

2155

2156

2157

Example 1: open a docbase

2158

Instructions sent from application to DCMS

2159

```
<uoml:OPEN xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" create="false"
```

2160

```
del_exist="false" path="c:\test.sep"/>
```

2161

Instructions returned from DCMS to application

2162

```
<!-- the string value "docbase001" is the opened docbase's handle for later use -->
```

2163

```
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
```

2164

```
  <boolVal name="SUCCESS" val="true"/>
```

2165

```
  <stringVal name="handle" val="docbase001"/>
```

2166

```
</uoml:RET>
```

2167

2168

Example 2 : get the root docset of the docbase (following example 1)

2169

Instructions sent from application to DCMS

2170

```
<!-- since each docbase has one and only one sub-object, to get the root docset is just to  
get the first sub-object of docbase whose handle is returned by example 1 -->
```

2171

2172

```
<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docbase001"
```

2173

```
usage="GET_SUB">
```

2174

```
  <pos val="0"/>
```

2175

```
</uoml:GET>
```

2176

Instructions returned from DCMS to application

2177

```
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
```

2178

```
  <boolVal name="SUCCESS" val="true"/>
```

2179 <stringValue name="handle" val="docset001"/>

2180 </uoml:RET>

2181

2182 **Example 3: get the number of sub-objects of the root docset (following example 2)**

2183 *Instructions sent from application to DCMS*

2184 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docset001"
2185 usage="GET_SUB_COUNT"/>

2186 *Instructions returned from DCMS to application*

2187 <!-- the return value of 3 indicates the root docset has 3 sub-objects -->

2188 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2189 <boolVal name="SUCCESS" val="true"/>

2190 <intVal name="sub_count" val="3"/>

2191 </uoml:RET>

2192

2193 **Example 4: get the third sub-object of the docset (following example 3)**

2194 *Instructions sent from application to DCMS*

2195 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docset001"
2196 usage="GET_SUB">

2197 <pos val="2"/>

2198 </uoml:GET>

2199 *Instructions returned from DCMS to application*

2200 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2201 <boolVal name="SUCCESS" val="true"/>

2202 <stringValue name="handle" val="doc001"/>

2203 </uoml:RET>

2204 **Examples 5: get the type of a object using the empty string as the name of the property (following example 4)**

2205 *Instructions sent from application to DCMS*

2206 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP"
2207 handle="doc001">

2208 <property name=""/>

2209 </uoml:GET>

2210 *Instructions returned from DCMS to application*

2211 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2212 <boolVal name="SUCCESS" val="true"/>

2213 <stringVal name="" val="DOC"/>

2214 </uoml:RET>

2215

2216 **Example 6: get the metadata of the document (following example 4)**

2217 *Instructions sent from application to DCMS*

2218 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP"
2219 handle="doc001">

2220 <property name="metainfo"/>

2221 </uoml:GET>

2222 *Instructions returned from DCMS to application*

2223 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2224 <boolVal name="SUCCESS" val="true"/>

2225 <compoundVal name="metainfo">

2226 <metalist>

2227 <meta key="title" val="UOML Part I"/>

2228 <meta key="author" val="UOML TC"/>

2229 </metalist>

2230 </compoundVal>

2231 </uoml:RET>

2232

2233 **Example 7: get page bitmap of a page**

2234 *Instructions sent from application to DCMS*

2235 <!-- the page object's handle is supposed to have already obtained of value "page001" in
2236 prior instructions(using GET) -->

2237 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PAGE_BMP"
2238 handle="page001">

```

2239     <disp_conf addr="c:\test.bmp" end_layer="8" format="bmp" output="FILE"
2240 resolution="640">
2241     <clip>
2242         <ellipse angle="45" center="10,20" xr="30" yr="40"/>
2243         <roundrect br="70,80" tl="50,60" xr="90" yr="100"/>
2244         <subpath data="s 214,193 1 368,193 1 368,298 1 214,298"/>
2245     </clip>
2246 </disp_conf>
2247 </uoml:GET>

2248 Instructions returned from DCMS to application

2249 <!-- the bmp format of page bitmap data has been saved in the file c:\test.bmp as requested
2250 -->

2251 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2252     <boolVal name="SUCCESS" val="true"/>
2253 </uoml:RET>
2254

```

2255 **Example 8 : get first layer of a page**

```

2256 Instructions sent from application to DCMS

2257 <!-- the page object's handle is supposed to have already obtained of value "page001" in
2258 prior instructions(using GET) -->

2259 <!-- since page has only layer objects as its sub-objects, get sub-objects is the same to
2260 get layer objects -->

2261 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="page001"
2262 usage="GET_SUB">
2263     <pos val="0"/>
2264 </uoml:GET>

2265 Instructions returned from DCMS to application

2266 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2267     <boolVal name="SUCCESS" val="true"/>
2268     <stringVal name="handle" val="layer001"/>
2269 </uoml:RET>

```

2270

2271 **Example 9: set a text object as the current object**

2272 *Instructions send from application to DCMS*

2273 *<!-- the text object's handle is supposed to have already obtained of value "text001" in*
2274 *prior instructions (using GET) -->*

2275 `<uoml:USE xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="text001"/>`

2276 *Instructions returned from DCMS to application*

2277 `<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">`

2278 `<boolVal name="SUCCESS" val="true"/>`

2279 `</uoml:RET>`

2280

2281 **Examples 10: get spaces property of a text object (following example 9)**

2282 *Instructions send from application to DCMS*

2283 `<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP">`

2284 `<property name="spaces"/>`

2285 `</uoml:GET>`

2286 *Instructions returned from DCMS to application*

2287 `<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">`

2288 `<boolVal name="SUCCESS" val="true"/>`

2289 `<stringVal name="spaces" val="50,55"/>`

2290 `</uoml:RET>`

2291

2292 **Example 11: insert a document into a docset (following example 2)**

2293 *Instructions send from application to DCMS*

2294 `<uoml:INSERT xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docset001">`

2295 `<xobj>`

2296 `<doc name="UOML part II">`

2297 `<metainfo>`

2298 `<meta key="author" val="alex"/>`

2299 </metainfo>

2300 </doc>

2301 </xobj>

2302 </uoml:INSERT>

2303 *Instructions returned from DCMS to application*

2304 <!-- the handle of the inserted document is returned for later use -->

2305 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2306 <boolVal name="SUCCESS" val="true"/>

2307 <stringVal name="handle" val="doc002"/>

2308 </uoml:RET>

2309

2310 **Example 12: delete the document inserted in the example above**

2311 *Instructions send from application to DCMS*

2312 <uoml:DELETE xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="doc002"/>

2313 *Instructions returned from DCMS to application*

2314 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2315 <boolVal name="SUCCESS" val="true"/>

2316 </uoml:RET>

2317

2318 **Example 13: use SYSTEM to save a docbase**

2319 *Instructions send from application to DCMS*

2320 <uoml:SYSTEM xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2321 <flush path="c:\test.sep"/>

2322 </uoml:SYSTEM>

2323 <!-- instructions returned from DCMS to application -->

2324 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2325 <boolVal name="SUCCESS" val="true"/>

2326 </uoml:RET>

2327

2328 **Example 14: close the docbase (following example 1)**

2329 *Instructions send from application to DCMS*

2330 <uoml:CLOSE xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docbase001"/>

2331 *instructions returned from DCMS to application*

2332 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2333 <boolVal name="SUCCESS" val="true"/>

2334 </uoml:RET>

2335 **End of informative text.**

2336
2337

2338

2339

2340
2341

2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376

Annex C.RELAX NG Representation of the UOML XML Schema

This annex is informative.

The following is a compact RELAX NG representation of the normative UOML XML Schema.

```
default namespace = ""
namespace ns1 = "urn:oasis:names:tc:uoml:xmlns:uoml:1.0"

start =
  (notAllowed
    | element ns1:OPEN {
      attribute path { xsd:string },
      attribute del_exist { xsd:boolean }?,
      attribute create { xsd:boolean }?
    })
  | (notAllowed
    | element ns1:RET {
      (element intVal { INT }
        | element floatVal { DOUBLE }
        | element timeVal { TIME }
        | element dateVal { DATE }
        | element dateTimeVal { DATETIME }
        | element durationVal { DURATION }
        | element stringVal { STRING }
        | element binaryVal { BINARY }
        | element compoundVal { COMPOUND }
        | element boolVal { BOOL }
        | element longVal {
          attribute val { xsd:long },
          attribute name { xsd:string }?
        }) *
      )
    })
  | (notAllowed
    | element ns1:SET {
      attribute handle { xsd:string }?,
      (element intVal { INT }
        | element floatVal { DOUBLE }
        | element timeVal { TIME }
        | element dateVal { DATE }
        | element dateTimeVal { DATETIME }
        | element durationVal { DURATION }
        | element stringVal { STRING }
        | element binaryVal { BINARY }
```

```

2377         | element compoundVal { COMPOUND }
2378         | element boolVal { BOOL })*
2379     })
2380 | (notAllowed
2381     | element ns1:GET {
2382         attribute handle { xsd:string }?,
2383         attribute usage { xsd:string },
2384         (element disp_conf {
2385             attribute addr { xsd:string },
2386             attribute output { xsd:string },
2387             attribute format { xsd:string }?,
2388             attribute resolution { xsd:int }?,
2389             attribute end_layer { xsd:int }?,
2390             element clip { PATH }?
2391         }
2392         | element pos {
2393             attribute val { xsd:int }
2394         }
2395         | element property {
2396             attribute name { xsd:string }
2397         })
2398     })
2399 | (notAllowed
2400     | element ns1:DELETE {
2401         attribute handle { xsd:string }?
2402     })
2403 | (notAllowed
2404     | element ns1:USE {
2405         attribute handle { xsd:string }
2406     })
2407 | (notAllowed
2408     | element ns1:INSERT {
2409         attribute pos { xsd:int }?,
2410         attribute handle { xsd:string }?,
2411         element xobj { COMPOUND }
2412     })
2413 | (notAllowed
2414     | element ns1:SYSTEM {
2415         element flush {
2416             attribute path { text }?,
2417             attribute handle { text }?
2418         }
2419     })
2420 | (notAllowed
2421     | element ns1:CLOSE {
2422         attribute handle { xsd:string }?
2423     })
2424 COMPOUND =
2425     (attribute name { xsd:string }?,

```

```

2426 ((notAllowed
2427 | element arc {
2428     attribute angle { xsd:float },
2429     attribute center { xsd:string },
2430     attribute end { xsd:string },
2431     attribute start { xsd:string },
2432     attribute clockwise { xsd:boolean }
2433 })
2434 | (notAllowed
2435 | element bezier {
2436     attribute end { xsd:string },
2437     attribute ctrl2 { xsd:string }?,
2438     attribute ctrl { xsd:string },
2439     attribute start { xsd:string }
2440 })
2441 | (notAllowed
2442 | element circle { CIRCLE })
2443 | (notAllowed
2444 | element cmd {
2445     attribute v2 {
2446         text
2447         # <data type="anySimpleType"/>
2448
2449     }?,
2450     attribute v1 {
2451         text
2452         # <data type="anySimpleType"/>
2453
2454     }?,
2455     attribute name {
2456         xsd:string "CHAR_WEIGHT"
2457         | xsd:string "CLIP_AREA"
2458         | xsd:string "COLOR_FILL"
2459         | xsd:string "CHAR_SIZE"
2460         | xsd:string "LINE_CAP"
2461         | xsd:string "SHADOW_LEN"
2462         | xsd:string "CHAR_STYLE"
2463         | xsd:string "RENDER_MODE"
2464         | xsd:string "CHAR_SLANT"
2465         | xsd:string "COLOR_LINE"
2466         | xsd:string "TEXT_DIR"
2467         | xsd:string "COLOR_TEXT"
2468         | xsd:string "GRAPH_MATRIX"
2469         | xsd:string "HOLLOW_BORDER"
2470         | xsd:string "POP_GS"
2471         | xsd:string "PUSH_GS"
2472         | xsd:string "LINE_WIDTH"
2473         | xsd:string "CHAR_DIR"
2474         | xsd:string "OUTLINE_WIDTH"

```

```

2475         | xsd:string "FILL_RULE"
2476         | xsd:string "EXT_MATRIX"
2477         | xsd:string "SHADOW_WIDTH"
2478         | xsd:string "RASTER_OP"
2479         | xsd:string "TEXT_MATRIX"
2480         | xsd:string "LINE_JOIN"
2481         | xsd:string "SHADOW_NEG"
2482         | xsd:string "SHADOW_ATL"
2483         | xsd:string "CHAR_ROTATE"
2484         | xsd:string "MITER_LIMIT"
2485         | xsd:string "COLOR_OUTLINE"
2486         | xsd:string "FONT"
2487         | xsd:string "IMAGE_MATRIX"
2488         | xsd:string "SHADOW_DIR"
2489         | xsd:string "OUTLINE_BORDER"
2490         | xsd:string "COLOR_SHADOW"
2491     },
2492     (element cliparea { PATH }
2493       | element matrix { MATRIX }
2494       | element rgb { COLOR_RGB })?
2495   ))
2496 | (notAllowed
2497   | element rgb { COLOR_RGB })
2498 | (notAllowed
2499   | element doc {
2500     attribute name { xsd:string },
2501     element metainfo { METALIST }
2502   })
2503 | (notAllowed
2504   | element docbase {
2505     attribute path { xsd:string },
2506     attribute name { xsd:string }
2507   })
2508 | (notAllowed
2509   | element docset {
2510     attribute name { xsd:string }
2511   })
2512 | (notAllowed
2513   | element ellipse { ELLIPSE })
2514 | (notAllowed
2515   | element embedfont { xsd:base64Binary })
2516 | (notAllowed
2517   | element fontlist { empty })
2518 | (notAllowed
2519   | element fontmap {
2520     attribute no { xsd:int },
2521     attribute name { xsd:string }
2522   })
2523 | (notAllowed

```

```

2524         | element image {
2525             attribute tl { xsd:string },
2526             attribute br { xsd:string },
2527             attribute type { xsd:string },
2528             attribute path { xsd:string }?,
2529             xsd:base64Binary
2530         })
2531     | (notAllowed
2532         | element layer { empty })
2533     | (notAllowed
2534         | element line {
2535             attribute end { xsd:string },
2536             attribute start { xsd:string }
2537         })
2538     | (notAllowed
2539         | element matrix { MATRIX })
2540     | (notAllowed
2541         | element meta { META })
2542     | (notAllowed
2543         | element metalist { METALIST })
2544     | (notAllowed
2545         | element page {
2546             attribute resolution { xsd:int },
2547             attribute height { xsd:float },
2548             attribute width { xsd:float }
2549         })
2550     | (notAllowed
2551         | element path { PATH })
2552     | (notAllowed
2553         | element rect { RECT })
2554     | (notAllowed
2555         | element roundrect { ROUNDRECT })
2556     | (notAllowed
2557         | element subpath { SUBPATH })
2558     | (notAllowed
2559         | element text {
2560             attribute spaces { xsd:string }?,
2561             attribute text { xsd:string },
2562             attribute encode { xsd:string },
2563             attribute origin { xsd:string }
2564         })
2565     | (notAllowed
2566         | element objstream { empty })))?),
2567     empty
2568 PATH =
2569     ((notAllowed
2570         | element subpath { SUBPATH })
2571     | (notAllowed
2572         | element rect { RECT })

```

```

2573     | (notAllowed
2574       | element circle { CIRCLE })
2575     | (notAllowed
2576       | element ellipse { ELLIPSE })
2577     | (notAllowed
2578       | element roundrect { ROUNDRECT }))**,
2579   empty
2580 METALIST =
2581   (notAllowed
2582     | element meta { META })*,
2583   empty
2584 COLOR_RGB =
2585   (attribute a { xsd:short }?,
2586     attribute b { xsd:short },
2587     attribute g { xsd:short },
2588     attribute r { xsd:short }),
2589   empty
2590 TIME =
2591   (attribute name { xsd:string }?,
2592     attribute val { xsd:time }),
2593   empty
2594 ELLIPSE =
2595   (attribute angle { xsd:float },
2596     attribute center { xsd:string },
2597     attribute yr { xsd:int },
2598     attribute xr { xsd:int }),
2599   empty
2600 SUBPATH =
2601   attribute data { xsd:string },
2602   empty
2603 INT =
2604   (attribute name { xsd:string }?,
2605     attribute val { xsd:int }),
2606   empty
2607 DURATION =
2608   (attribute name { xsd:string }?,
2609     attribute val { xsd:duration }),
2610   empty
2611 ROUNDRECT =
2612   (attribute br { xsd:string },
2613     attribute tl { xsd:string },
2614     attribute yr { xsd:int },
2615     attribute xr { xsd:int }),
2616   empty
2617 DATE =
2618   (attribute name { xsd:string }?,
2619     attribute val { xsd:date }),
2620   empty
2621 BINARY =

```

```

2622     (attribute name { xsd:string }?,
2623     attribute val { xsd:base64Binary })),
2624     empty
2625 STRING =
2626     (attribute name { xsd:string }?,
2627     attribute val { xsd:string })),
2628     empty
2629 DOUBLE =
2630     (attribute name { xsd:string }?,
2631     attribute val { xsd:double })),
2632     empty
2633 BOOL =
2634     (attribute name { xsd:string }?,
2635     attribute val { xsd:boolean })),
2636     empty
2637 CIRCLE =
2638     (attribute center { xsd:string },
2639     attribute radius { xsd:int })),
2640     empty
2641 META =
2642     (attribute val { xsd:string },
2643     attribute key { xsd:string })),
2644     empty
2645 MATRIX =
2646     (attribute f32 { xsd:float },
2647     attribute f31 { xsd:float },
2648     attribute f22 { xsd:float },
2649     attribute f21 { xsd:float },
2650     attribute f12 { xsd:float },
2651     attribute f11 { xsd:float })),
2652     empty
2653 RECT =
2654     (attribute br { xsd:string },
2655     attribute tl { xsd:string })),
2656     empty
2657 DATETIME =
2658     (attribute name { xsd:string }?,
2659     attribute val { xsd:dateTime })),
2660     empty

```

2661

2662

2663 **End of informative text.**

2664

2666

Annex D.Acknowledgements

2667

2668

2669 **This annex is informative.**

2670 The following individuals have participated in the creation of this specification and are gratefully acknowledged:

2671

2672 **Participants:**

2673 Alex Wang, Sursen Corporation

2674 Xu Guo, Sursen Corporation

2675 Ningsheng Liu, Sursen Corporation

2676 Allison Shi, Sursen Corporation

2677 Stephen Green, Individual

2678 Kaihong Zou, Sursen Corporation

2679 Pine Zhang, UOML Alliance

2680 Mendy Liu, UOML Alliance

2681 Joel Marcey, Sursen Corporation

2682 Andy Li, Changfeng Open Standards Platform Software Alliance

2683 Charles H. Schulz, Ars Aperta

2684 Lin Cheng, Beijing Redflag CH2000 Software Co. Ltd.

2685 Liwei Wang, Sursen Corporation

2686

2687 **End of informative text.**