

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

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

OASIS Working Draft

25 March 2011

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>
Peter Junge, Sursen Corporation <peter@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.

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

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

50 **Status:**

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

55 Technical Committee members should send comments on this specification to the Technical
56 Committee’s email list. Others should send comments to the Technical Committee by using the “Send A
57 Comment” button on the Technical Committee’s web page at [http://www.oasis-](http://www.oasis-open.org/committees/uoml-x/)
58 [open.org/committees/uoml-x/](http://www.oasis-open.org/committees/uoml-x/).

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

62 The non-normative errata page for this specification is located at [http://www.oasis-](http://www.oasis-open.org/committees/uoml-x/)
63 [open.org/committees/uoml-x/](http://www.oasis-open.org/committees/uoml-x/).

Notices

65 Copyright © OASIS® 2010. All Rights Reserved.

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

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

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

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

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

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

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

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

106

107

Table of Contents

1. Introduction	7
1.1 Terminology	8
1.2 Scope	11
1.3 Notational Conventions	12
1.4 Acronyms and Abbreviations	13
1.5 General Description	14
1.6 Overview	15
1.7 Normative References	16
1.8 Non-Normative References	18
2. Abstract Document Model	19
2.1 Overview	19
2.2 Docbase	20
2.3 Docset	20
2.4 Document	20
2.5 Font	21
2.6 Page	21
2.7 Layer	21
2.8 Object Stream	22
2.9 Graphics Object	22
2.10 Command Object	22
2.11 UML Diagram of UOML	22
2.12 Page Rendering Model	23
3. UOML Instructions	24
3.1 OPEN	24
3.2 CLOSE	25
3.3 USE	25
3.4 GET	26
3.5 SET	29
3.6 INSERT	29
3.7 DELETE	30
3.8 SYSTEM	31
3.9 RET	31
4. UOML Objects	33
4.1 Logical Coordinate System and Units	33
4.2 Color Model	33
4.3 Graphics State	34
4.4 DOCBASE	34
4.5 DOCSET	34
4.6 DOC	34
4.6.1 Metadata	34
4.7 FONT DEFINITION	35
4.7.1 FONTLIST	35
4.7.2 FONTMAP	35
4.7.3 EMBEDFONT	35

4.8	PAGE.....	36
4.9	LAYER.....	36
4.10	OBJSTREAM.....	36
4.11	Graphics Objects.....	36
4.11.1	ARC.....	36
4.11.2	BEZIER.....	37
4.11.3	CIRCLE.....	37
4.11.4	ELLIPSE.....	37
4.11.5	IMAGE.....	37
4.11.6	LINE.....	38
4.11.7	RECT.....	38
4.11.8	ROUNDRECT.....	38
4.11.9	SUBPATH.....	39
4.11.10	PATH.....	39
4.11.11	TEXT.....	40
4.11.12	Coordinate and subpath Encoding Rules.....	41
4.12	Command Object.....	42
4.12.1	CMD.....	42
4.12.2	Values for CMD's 'name' property.....	43
4.12.3	Definition of Referenced Type.....	57
4.13	Default Value of Graphics State.....	58
4.14	Definition of Parameter Data Types.....	59
4.14.1	INT.....	60
4.14.2	DOUBLE.....	60
4.14.3	LONG.....	60
4.14.4	DATE.....	60
4.14.5	TIME.....	60
4.14.6	DATETIME.....	60
4.14.7	DURATION.....	61
4.14.8	STRING.....	61
4.14.9	BINARY.....	61
4.14.10	BOOL.....	61
4.14.11	COMPOUND.....	61
4.15	Data Ranges.....	62
5.	Conformance.....	63
5.1.1	DCMS Conformance.....	63
5.1.2	Application Conformance.....	63
Annex A.	UOML XML Schema.....	64
Annex B.	Detailed UOML Examples.....	76
Annex C.	RELAX NG Representation of the UOML XML Schema.....	83
Annex D.	Acknowledgements.....	91

1. Introduction

2 This text is informative

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

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

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

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

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

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

25 End of informative text

26

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
61 object, 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*]

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.

112 This standard does not define any binding for the operations on the UOML document model. Such bindings are
113 implementation-defined or will be defined in further standards focusing UOML and related technologies.

114

115 **1.3 Notational Conventions**

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

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

119

120 **1.4 Acronyms and Abbreviations**

121 **This clause is informative**

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

123 DCMS — Docbase Management System

124 IEC — the International Electrotechnical Commission

125 ISO — the International Organization for Standardization

126 UOML — Unstructured Operation Markup Language

127 W3C — World Wide Web Consortium

128 **End of informative text**

129

130 1.5 General Description

131 This OASIS standard is divided into the following subdivisions:

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

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

138 The following items form the normative pieces of this OASIS standard:

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

140 The following items form the informative pieces of this OASIS standard:

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

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

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

151

152 1.6 Overview

153 **This clause is informative**

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

157 UOML is to unstructured documents as SQL (Structured Query Language) is to structured data. UOML is
158 expressed using standard XML via an instance of an XML schema. UOML handles fixed-layout documents and
159 its associated information (e.g., metadata, security rights, etc.) Fixed-layout- documents are two-dimensional
160 and contain static paging information (i.e., information that can be recorded on traditional paper). Thus, the
161 document stores fixed-layout 2D static information that describes the visual appearance. It does not store
162 dynamic graphic elements such as animation or interactive forms.

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

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

172 The main difference between UOML and other standards for layout-oriented document representation (e.g.
173 PDF, SVG, HTML+XSL-FO) is that UOML is an interface standard, rather than a storage standard.

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

176

177 **End of informative text**

178

179

180 1.7 Normative References

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

184

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

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

188 **[RGB]** IEC 61966-2-1: 1999: Multimedia systems and equipment — Colour measurement and management —
189 Part 2-1: Colour management — Default RGB colour space — sRGB. International Electrotechnical
190 Commission, 1999. ISBN 2-8318-4989-6 as amended by Amendment A1:2003.

191 **[DATE]** ISO 8601:2004, *Data elements and interchange formats – Information Interchange – Representation of*
192 *dates and times*.

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

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

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

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

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

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

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

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

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

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

210 **[SVG]** *Scalable Vector Graphics (SVG) 1.1 Specification*, W3C,2003, [http://www.w3.org/TR/2003/REC-](http://www.w3.org/TR/2003/REC-SVG11-20030114/)
211 [SVG11-20030114/](http://www.w3.org/TR/2003/REC-SVG11-20030114/)

212 **[Unicode]** *The Unicode Standard*, 5th edition, The Unicode Consortium, Addison-Wesley Professional,
213 ISBN 0321480910, <http://www.unicode.org/versions/Unicode5.0.0/>

214 **[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)
215 [schema-errata.xsd](http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-part1-v1.0-schema-errata.xsd)

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

218 **[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/)
219 [xml-names11-20060816/](http://www.w3.org/TR/2006/REC-xml-names11-20060816/)

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

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

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

226

227

228 **1.8 Non-Normative References**

229 **This clause is informative.**

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

231 **End of informative text.**

232

2. Abstract Document Model

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

239

2.1 Overview

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

245

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

250

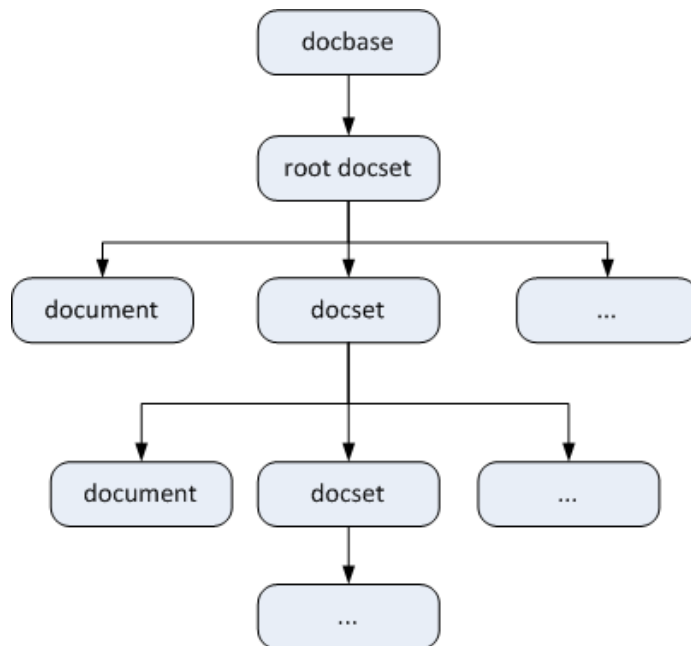


Figure1. UOML Abstract document Model 1

251
252

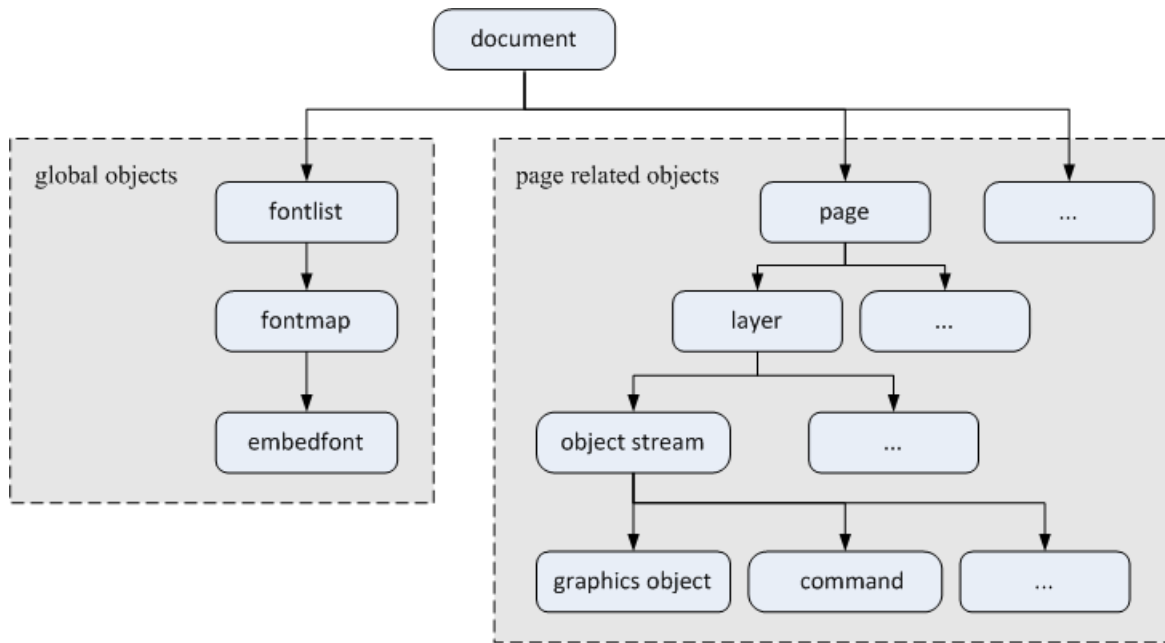


Figure 2. UOML Abstract Document Model 2

253

254 The following clauses provide a description of each object type.

255 2.2 Docbase

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

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

260 **Sub-object:** docset.

261 2.3 Docset

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

264 **Sub-object:** document, docset.

265 2.4 Document

266 The document object is the root node of document information (see Figure 2). A document contains static
 267 information for fixed-layout 2D documents [*Note: In future UOML parts or future versions of this part, other
 268 types of document information may be supported, including audio/video, 3D information, etc. end note*]. A
 269 single document has zero to multiple pages. In this specification, a document object is specified using DOC.
 270 (§4.5).

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

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

274

275 2.5 Font

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

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

282 2.6 Page

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

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

288

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

290

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

293

294 **Sub-object:** layer.

295 2.7 Layer

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

299

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

308

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

310

311 **Sub-object:** object stream.

312 **2.8 Object Stream**

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

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

318 **Sub-object:** graphics object, command object.

319 **2.9 Graphics Object**

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

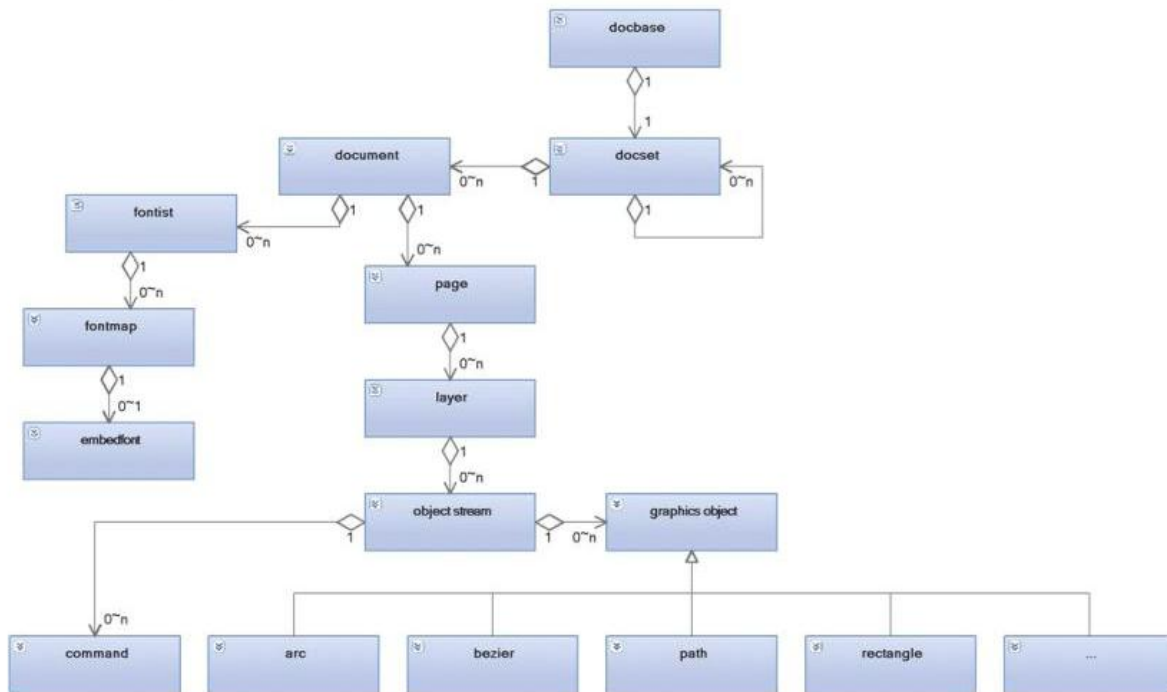
323 **2.10 Command Object**

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

327 **2.11 UML Diagram of UOML**

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

330



331

332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347

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.

A DCMS engine should stop and dismiss rendering on any error occurring during the rendering process.

348

3. UOML Instructions

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

351

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

3.1 OPEN

Semantics:

357 OPEN creates or opens a docbase.

Properties:

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

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

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

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

Return value:

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

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

[Example:

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

379

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

381

382 Return element if OPEN succeeds:

383

```
384 <uoml:RET>
```

```
385     <boolVal name="SUCCCESS" val="true"/>
```

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



```
387     </uoml:RET>
388
389     Return element if OPEN fails:
390
391     <uoml:RET>
392         <boolVal name="SUCCCESS" val="false"/>
393         <stringVal name="ERR_INFO" val="required resource not available"/>
394     </uoml:RET>
```

395
396 *end example]*

397 **3.2 CLOSE**

398 **Semantics:**

399 CLOSE closes a docbase

400 **Properties:**

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

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

403 **Return value:**

404 Defined by RET

405 [*Example:*

406 Close a docbase.

407

```
408     <uoml:CLOSE handle="db_handle_XXXXXX"/>
```

409

410 *end example]*

411 **3.3 USE**

412 **Semantics:**

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

416 **Properties:**

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

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

419 **Return value:**

420 Defined by RET

421 [*Example:*

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

423

```
424     <uoml:USE handle="obj_handle_XXXXXX"/>
```

425 *end example]*

426 3.4 GET

427 Semantics:

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

430 Properties:

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

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

436 Sub-elements:

437 *pos*: used when usage=GET_SUB.

438 Property of this sub-element:

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

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

441

442 *property*: used when usage=GET_PROP.

443 Property of this sub-element:

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

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

447

448 *disp_conf*: used when usage=GET_PAGE_BMP.

449 Properties of this sub-element:

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

452 *resolution*: represents resolution of bitmap

453 *format*: represents the bitmap format. Valid values are "bmp", representing the uncompressed
454 BMP format and "svg", representing the Scalable Vector Graphics Format.

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

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

458 Sub-element of this sub-element:

459 *clip*: represents clip area for output, PATH type.

460

461 Usage value / Return value:

462 The return value is based on the usage value:

463

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

469 *[Example:*

470

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

472

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

474

475 RET instruction returns the number:

476

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

481

482 *end example]*

483

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

488 *[Example:*

489

490 Get a specific sub-object handle:

491

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

495

496 RET instruction returns the handle of the sub-object:

497

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

502

503 *end example]*

504

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

512 description of the object without the namespace prefix.

513 *[Example:*

514
515 Get specific property of the object
516
517 `<uoml:GET handle="obj_handle_xxxxx" usage="GET_PROP">`
518 `<property name="start"/>`
519 `</uoml:GET>`

520
521 RET instruction returns the start property, which is a coordinate:

522
523 `<uoml:RET>`
524 `<boolVal name="SUCCESS" val="true"/>`
525 `<stringVal name="start" val="200,300"/>`
526 `</uoml:RET>`

527
528 *end example]*

- 529
- 530 ○ GET_PAGE_BMP: If the usage is GET_PAGE_BMP, this indicates to get the specific page bitmap.
 - 531 In this case, GET shall contain the sub-element 'disp_conf'. The requested bitmap should be
 - 532 placed/returned where the 'addr' and 'output' property of the 'disp_conf' element is specified.

533 *[Example:*

534
535 Get specific page's bitmap
536
537 `<uoml:GET handle="page_obj_handle_xxx" usage="GET_PAGE_BMP">`
538 `<disp_conf format="bmp" output="FILE" end_layer="1" resolution="600"`
539 `path="/home/admin/output/page.bmp">`
540 `<clip>`
541 `<subpath data="s 0,0 1 3000,0 1 3000, 5000 1 0, 5000 1 0,0"/>`
542 `</clip>`
543 `</disp_conf>`
544 `</uoml:GET>`

545
546 *end example]*

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

548
549 *[Example:*

550
551 `<uoml:RET>`
552 `<boolVal name="SUCCESS" val="false"/>`
553 `<stringVal name="ERR_INFO" val="disk full"/>`
554 `</uoml:RET>`

555
556 *end example]*

557

558 3.5 SET

559 Semantics:

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

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

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

563 Properties:

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

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

566 Sub-element:

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

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

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

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

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

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

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

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

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

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

577 Return value:

578 defined by RET.

579 [Example:

580 Set specific object's angle property.

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

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

583 `</uoml:SET>`

584 *end example]*

585

586 3.6 INSERT

587 Semantics:

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

589 Properties:

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

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

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

597 **Sub-element:**

598 *xobj*: xml expression of the sub-object.

599 **Return value:**

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

602

603 [*Example*:

604 Insert text data

605

```
606 <uoml:INSERT pos="1"/>  
607 <xobj>  
608 <text origin="100, 200" encode="ASCII" text="UOML"  
609 spaces="20,20,20"/>  
610 </xobj>  
611 </uoml:INSERT>
```

612

613

614 *end example*]

615

616

617 [*Example*:

618

619 Insert a layer

620

```
621 <uoml:INSERT handle="page_obj_handle_XXXXXX">  
622 <xobj>  
623 <layer/>  
624 </xobj>  
625 </uoml:INSERT>
```

626 *end example*]

627

628 3.7 DELETE

629 **Semantics:**

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

632 **Properties:**

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

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

636 **Return value:**

637 Defined by RET

638 [*Example:*

639 Delete an object

640

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

642 *end example]*

643

644 **3.8 SYSTEM**

645 **Semantics:**

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

648 **Properties:**

649 N/A

650 **Sub-element:**

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

653 **Return value:**

654 Defined by RET

655 [*Example:*

656 Save the docbase example.sep

657

658 `<uoml:SYSTEM>`
659 `< flush handle="docbase_handle_xxxxx"`
660 `path="/home/admin/storage/example.sep"/>`
661 `</uoml:SYSTEM>`

662 *end example]*

663

664 **3.9 RET**

665 **Semantics:**

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

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

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

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

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

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

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

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

681

682 **Properties:** N/A

683

684 **Sub-element:**

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

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

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

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

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

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

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

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

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

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

695

696 [*Example*:

697 Return two values.

698 <uoml:RET>

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

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

701 </uoml:RET>

702 *end example*]

703

4. UOML Objects

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

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

708

4.1 Logical Coordinate System and Units

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

714

715 The length of the units along each axis depends on the resolution property of the page. If the resolution of a
716 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
717 resolution of the page.

718

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

720

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

4.2 Color Model

724 UOML uses sRGB color space [RGB] to describe color. A color has red(R), green(G) and blue(B) components,
725 the value of each component falls within the data range from zero to 255. An 8-bit alpha channel is used for
726 the purpose of compositing images, so there are "r", "g", "b" and "a" attributes for the XML description of
727 color. The value of the resulting color when color Value1 with an alpha value of α is drawn over an opaque
728 background of color Value0 is given by:

729
$$\text{Value} = (1 - \alpha)\text{Value0} + \alpha\text{Value1}$$

730

731 4.3 Graphics State

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

736 4.4 DOCBASE

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

738 **Properties:**

739 *name*: name of docbase.

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

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

743 4.5 DOCSET

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

745 **Properties:**

746 *name*: name of docset.

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

748 4.6 DOC

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

750 **Properties:**

751 *name*: name of document.

752 **Sub-elements:**

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

754

755 4.6.1 Metadata

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

760 **4.6.1.1 METALIST**

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

762 **Properties:** N/A

763 **Sub-elements:**

764 *meta*: META type.

765 **4.6.1.2 META**

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

767 **Properties:**

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

771 *val*: Unicode character string value representing the value of metadata.

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

773

774 **4.7 FONT DEFINITION**

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

776 **4.7.1 FONTLIST**

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

778 **Properties:** N/A

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

780 **4.7.2 FONTMAP**

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

782 **Properties:**

783 *name*: name of the font

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

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

788 **4.7.3 EMBEDFONT**

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

792 **Properties:** N/A

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

794 4.8 PAGE

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

796 **Properties:**

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

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

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

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

802 4.9 LAYER

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

804 **Properties:** N/A

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

806 4.10 OBJSTREAM

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

808 **Properties:** N/A

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

810 4.11 Graphics Objects

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

813

814

815 4.11.1 ARC

816 **Semantics:**

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

819 **Properties:**

820 *start*: starting position of the arc.

821 *end*: ending position of the arc.

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

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

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

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

829 4.11.2 BEZIER

830 Semantics:

831 A second-order or third-order Bezier curve. A Bezier curve is specified using three or four properties:
832 the starting point, the ending point, one control point and, optionally, a second control point. A
833 second-order Bezier curve is specified when only one control point is used. A third-order Bezier curve is
834 specified when a second control point is used.

835 Properties:

836 *start*: starting point of the Bezier curve.
837 *ctrl*: the first control point of the Bezier curve.
838 *ctrl2*: the optional second control point of the Bezier curve.
839 *end*: ending point of the Bezier curve.

840 Sub-elements: N/A

841 4.11.3 CIRCLE

842 Semantics:

843 A circle, specified by a center and radius.

844 Properties:

845 *center* : coordinate of the circle center.
846 *radius*: positive integer value representing the radius of the circle.

847 Sub-elements: N/A

848 4.11.4 ELLIPSE

849 Semantics:

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

851 Properties:

852 *center*: coordinates of ellipse center.
853 *xr*: positive integer value representing the length of the x-radius.
854 *yr*: positive integer value representing the length of the y-radius.
855 *angle*: inclination from coordinate system's x-axis to ellipse's x-axis. It is specified using a radian value
856 of type xs:float. A positive value represents counterclockwise and a negative value represents clockwise.

857 Sub-elements: N/A

858 4.11.5 IMAGE

859 Semantics:

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

866 **Properties:**
867 *tl*: coordinates of the top-left corner of the image
868 *br*: coordinates of the bottom-right corner of the image
869 *type*: image type, possible values include "bmp", "png", "jpeg", "jbig", "tiff", representing BMP, PNG,
870 JPEG, JBIG, TIFF images respectively.
871 *path*: path of the image file. This is an optional property, but if present, the content of IMAGE element
872 should be left blank; otherwise the content of IMAGE element contains the base64 encoded raw image
873 data.

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

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

876 **4.11.6 LINE**

877 **Semantics:**

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

879 **Properties:**

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

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

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

883

884 **4.11.7 RECT**

885 **Semantics:**

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

887 **Properties:**

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

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

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

891

892 **4.11.8 ROUNDRECT**

893 **Semantics:**

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

895 **Properties:**

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

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

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

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

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

901 4.11.9 SUBPATH

902 **Semantics:**

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

905 **Properties:**

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

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

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

```
914 <uoml:INSERT pos="2" handle="vs03">  
915 <xobj>  
916 <path>  
917 <subpath data="s 214,193 l 368,193 l 368,298 l 214,298"/>  
918 <subpath data="s 417,206 B 417,186 426,167 435,167 B 443,167 452,230 452,293"/>  
919 </path>  
920 </xobj>  
921 </uoml:INSERT>
```

922
923 *end example*].

924

925 4.11.10 PATH

926 **Semantics:**

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

930 **Properties:** N/A

931 **Sub-elements:**

932 *circle*: CIRCLE type, defines a circle.
933 *ellipse*: ELLIPSE type, defines an ellipse.
934 *rect*: RECT type, defines a rectangle.
935 *roundrect*: ROUNDRECT type, defines a rectangle with round corners.
936 *subpath*: SUBPATH type, defines a subpath.

937

938 [*Example*: The following example demonstrates a PATH consisting of two sub elements: a rectangle and a
939 circle.

```

940     <uoml:INSERT pos="4">
941         <xobj>
942             <path>
943                 <circle center="167,251" radius="70" />
944                 <rect tl="124,135" br="345,257"/>
945             </path>
946         </xobj>
947     </uoml:INSERT>

```

948

949 *end example*].

950

951 4.11.11 TEXT

952 Semantics:

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

954 Properties:

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

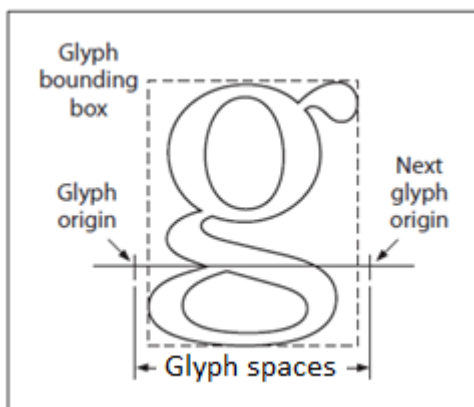
957 *encode*: character set or encoding of text data. The valid value for this property should be one of the
958 character encodings registered (as charsets) with the Internet Assigned Numbers Authority [IANA-
959 CHARSETS], otherwise it should use names starting with an x- prefix.

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

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

963

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



968

969 Figure 4. spaces of text

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

974

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

976

977 **4.11.12 Coordinate and subpath Encoding Rules**

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

980

981 **Coordinate encoding rules**

982

```
983 coord = coordx, [blank] , ',' , [blank] , coordy ;  
984 coordx = number ;  
985 coordy = number ;
```

986

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

989

990 **Path encoding rules**

991

992

```
993 path = start , { blank , ( line | bezier2 | bezier3 | arc ) } ;  
994 start = 's' , blank , coord ;  
995 line = 'l' , blank , coord ;  
996 bezier2 = 'b' , blank , coord , blank , coord ;  
997 bezier3 = 'B' , blank , coord , blank , coord , blank , coord ;  
998 arc = 'a' , blank , clockwise , blank , angle , blank , coord , blank , coord ;  
999 clockwise = 'true' | 'false' ;  
1000 angle = float ;  
1001 number = [ '-' ] , digit , { digit } ;  
1002 float = number [ , '.' , { digit } ] [ , ( 'e' | 'E' ) , [ '+' | '-' ] , digit , { digit } ] ;  
1003 digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;  
1004 blank = ' ' , { ' ' } ;
```

1005

1006 **Semantics**

1007 "coord" represents coordinates.

1008 "start" represents the start point of the subpath.

1009 "line" represents a line segment.

1010 "bezier2" represents a second-order Bezier curve.

1011 "bezier3" represents third-order Bezier curve.

1012 "blank" represents one or many blanks or an equivalent whitespace character, such as a tab, carriage
1013 return or a new line.

1014
1015 In the definition of "line", the "coord" represents the ending point.

1016 In the definition of "bezier", the two "coord" are for the control point and the ending point.

1017 In the definition of "bezier3", the three "coord" are for the control point 1, control point 2 and
1018 ending point.

1019 In the definition of "arc", the two "coord" are the center and end points.

1020 [Note: The start point of each item is the previous end point. *end note*]

1021

1022 4.12 Command Object

1023 A command object is used for modifying the graphics, such as text size, typeface and color.

1024 4.12.1 CMD

1025 **Semantics:** XML description of command objects.

1026 **Properties:**

1027 name: name of the command. [Note: §4.12.2 provides possible values for this property. *end note*]

1028 v1: optional command value.

1029 v2: optional command value.

1030 **Sub-elements:**

1031 *rgb*: a COLOR_RGB value (§4.12.3.1), used when 'name' is one of COLOR_LINE, COLOR_FILL,
1032 COLOR_SHADOW, COLOR_OUTLINE or COLOR_TEXT.

1033 *matrix*: a MATRIX value (§4.12.3.2), used when 'name' is one of TEXT_MATRIX, IMAGE_MATRIX,
1034 GRAPH_MATRIX or EXT_MATRIX.

1035 *cliparea*: a PATH value, used when 'name' is CLIP_AREA.

1036 **Sub-objects:** N/A

1037 [Example:

1038

```
1039 <uoml:INSERT pos="2" handle="vs03">
```

```
1040 <xobj>
```

```
1041 <cmd name="COLOR_LINE" >
```

```
1042 <rgb r="128" g="3" b="255" a="120"/>
```

1043 </cmd>
1044 </xobj>
1045 </uoml:INSERT>

1046

1047 *end example]*

1048

1049 [*Example:*

1050

```
1051     <uoml:INSERT pos="2" handle="vs03">  
1052         <xobj>  
1053             <cmd name="LINE_CAP" v1="END_BUT"/>  
1054         </xobj>  
1055     </uoml:INSERT>
```

1056

1057 *end example]*

1058

1059 [*Example:*

```
1060     <uoml:INSERT pos="2" handle="vs03">  
1061         <xobj>  
1062             <cmd name="TEXT_MATRIX">  
1063                 <matrix f11="2" f12="0" f21="0" f22="1.5" f31="10" f32="20"/>  
1064             </cmd>  
1065         </xobj>  
1066     </uoml:INSERT>
```

1067

1068 *end example]*

1069 **4.12.2 Values for CMD's 'name' property**

1070 This clause describes the values that may be used for CMD's 'name' property, and which properties and sub-
1071 elements may be used for each valid 'name' value. [*Example:* If the CMD's 'name' property is 'COLOR_LINE',
1072 then CMD's sub-element is 'rgb'. *end example]*

1073

1074 In order to simplify the parsing process, properties (command values) within command objects all have a
1075 general name called v1 (and v2 if there is a second property) no matter what they represent.

1076 **4.12.2.1 COLOR_LINE**

1077 **Semantics:** Set the current line color

1078 **Properties:** N/A

1079 **Sub-elements:**

1080 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to stroke lines and
1081 curves.

1082 **4.12.2.2** COLOR_FILL

1083 **Semantics:** Set the current fill color

1084 **Properties:** N/A

1085 **Sub-elements:**

1086 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to fill an area.

1087 **4.12.2.3** COLOR_SHADOW

1088 **Semantics:** Set the current character shadow color

1089 **Properties:** N/A

1090 **Sub-elements:**

1091 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to draw the shadow of
1092 characters.

1093 **4.12.2.4** COLOR_OUTLINE

1094 **Semantics:** Set the current character outline color

1095 **Properties:** N/A

1096 **Sub-elements:**

1097 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to draw the outline of
1098 characters.

1099 **4.12.2.5** COLOR_TEXT

1100 **Semantics:** Set the current text color

1101 **Properties:** N/A

1102 **Sub-elements:**

1103 *rgb*: element of the COLOR_RGB (§4.11.3.1) type. RGB specifies the color used to draw characters.

1104 **4.12.2.6** LINE_WIDTH

1105 **Semantics:** set the current line width/thickness

1106 **Properties:**

1107 *v1*: a positive floating point number, representing the width of the line.

1108 **Sub-elements:** N/A

1109 **4.12.2.7** LINE_CAP

1110 **Semantics:** Set the current line cap style

1111 **Properties:**

1112 *v1*: a character string, representing the line cap style. Possible values for this property are END_BUT,
1113 END_ROUND and END_SQUARE.

1114 END_BUT: the stroke shall be squared off at the endpoint of the path. There shall be no projection
1115 beyond the end of the path.

1116
1117



1118
1119

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

1120
1121



1122
1123

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

1124
1125



1126 **Sub-elements:** N/A

1127 4.12.2.8 [LINE_JOIN](#)

1128 **Semantics:** Set the current line join style

1129 **Properties:**

1130 *v1*: a character string, representing the line join style. Possible values for this property are JOIN_MITER,
1131 JOIN_BEVEL and JOIN_ROUND

1132

1133 JOIN_MITER: the outer edges of the strokes for the two segments shall be extended until they meet at
1134 an angle. If the segments meet at too sharp an angle as measured by the current miter length
1135 maximum, the value JOIN_BEVEL shall be used instead.



1136
1137

1138 JOIN_BEVEL: the two segments shall be finished with END_BUT and the resulting notch beyond the end
1139 of the segments shall be filled with a triangle.



1140

1141

1142

1143

1144

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



1145

1146

1147

Sub-elements: N/A

1148

4.12.2.9 MITER_LIMIT

1149

1150

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

1151

1152

Properties:

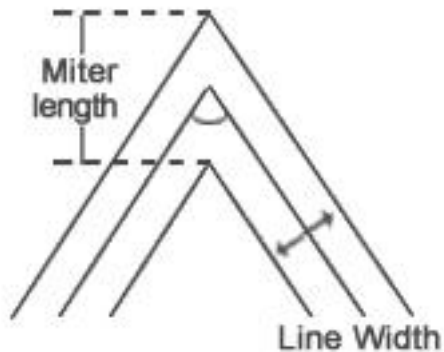
1153

v1: a positive floating point number, representing the maximum ratio.

1154

Sub-elements: N/A

1155



1156

1157

4.12.2.10 FILL_RULE

1158

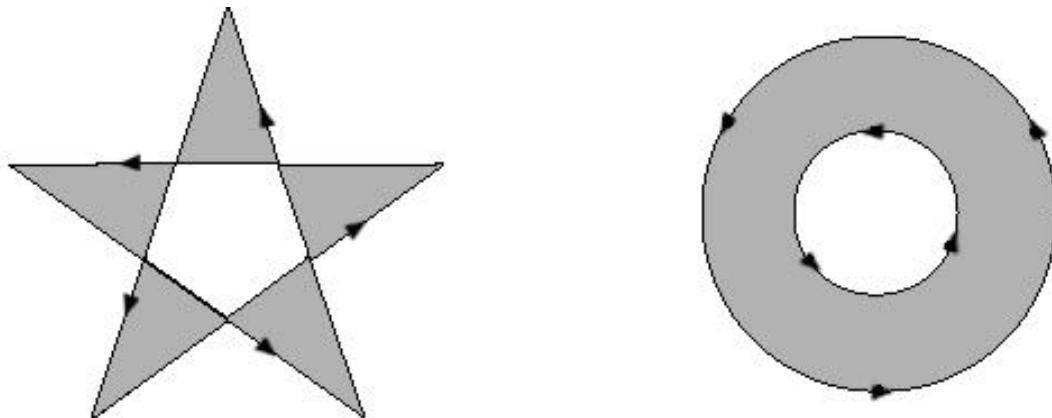
Semantics: Set the current fill rules

1159 **Properties:**

1160 *v1*: a character string, representing the fill rule. The possible values for this property are
1161 RULE_EVENODD and RULE_WINDING.

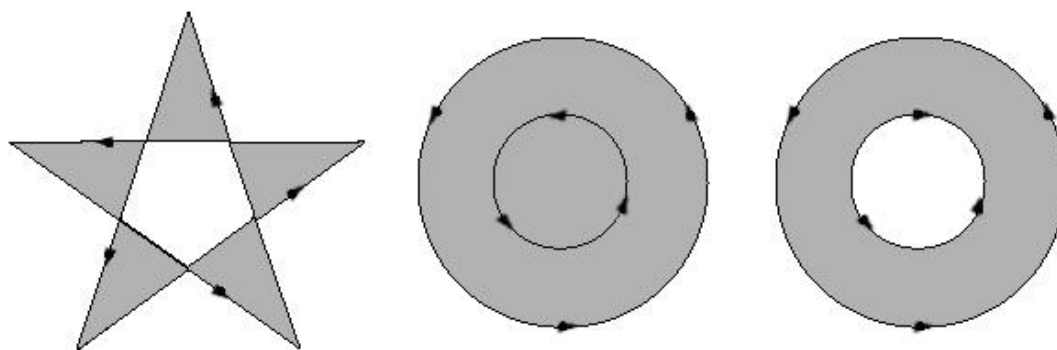
1162

1163 RULE_EVENODD: Specifies that areas are filled according to the even-odd parity rule. According to this
1164 rule, it can be determined whether a test point is inside or outside a closed curve as follows: Draw a ray
1165 from the test point in any direction and count the number of path segments that cross the ray,
1166 regardless of the direction. If the number is odd, the point is inside; if the number is even, the point is
1167 outside.



1168

1169 RULE_WINDING: Specifies that areas are filled according to the nonzero winding rule. According to this
1170 rule, it can be determined whether a test point is inside or outside a closed curve as follows: draw a ray
1171 from that point to infinity in any direction and examine the places where a segment of the path crosses
1172 the ray. Starting with a count of 0, the rule adds 1 each time a curve segment crosses the ray from left
1173 to right and subtracts 1 each time a segment crosses from right to left. After counting all the crossings,
1174 if the result is 0, the point is outside the path; otherwise, it is inside.



1175

1176

1177 **Sub-elements:** N/A

1178 **Note:**

1179 **4.12.2.11 RENDER_MODE**

1180 **Semantics:** Set the current render mode (line, fill, clip, or their combination)

1181 **Properties:**

1182 `v1`: a character string, representing the render mode. The possible values for this property are LINE,
1183 FILL, CLIP, or some combination of the three, with values separated by a comma.

1184 LINE: draw a line along the path.

1185 FILL: draw the entire region enclosed by the path.

1186 CLIP: current clip area will be set as the intersection of the next path graphics and current clip area.

1187 **Sub-elements:** N/A

1188 4.12.2.12 RASTER_OP

1189 **Semantics:** Set the current raster operation.

1190 **Properties:**

1191 `v1`: a character string, representing the raster operation. The possible values for this property are
1192 ROP_COPY, ROP_N_COPY, ROP_RESET, ROP_SET, ROP_NOP, ROP_REV, ROP_AND, ROP_AND_N,
1193 ROP_N_AND, ROP_N_AND_N, ROP_OR, ROP_OR_N, ROP_N_OR, ROP_N_OR_N, ROP_XOR, and
1194 ROP_EOR. In the following, 'pixel_color' represents the color after a raster operation; 'src' is the
1195 currently used color; 'dest' is the current color of the destination bitmap to be drawn upon; '&' is
1196 bitwise AND; '|' is bitwise OR; '^' is bitwise XOR; and '~' is bitwise NOT, which has the highest priority
1197 over the other logical operators.

1198

1199 ROP_COPY: pixel_color = src

1200 ROP_N_COPY: pixel_color = ~src

1201 ROP_RESET: pixel_color = 0 (all bits of pixel_color are set zero)

1202 ROP_SET: pixel_color = 1 (all bits of pixel_color are set 1)

1203 ROP_NOP: pixel_color = dest

1204 ROP_REV: pixel_color = ~dest

1205 ROP_AND: pixel_color = src & dest

1206 ROP_AND_N: pixel_color = src & ~dest

1207 ROP_N_AND: pixel_color = ~src & dest

1208 ROP_N_AND_N: pixel_color = ~src & ~dest

1209 ROP_OR: pixel_color = src | dest

1210 ROP_OR_N: pixel_color = src | ~dest

1211 ROP_N_OR: pixel_color = ~src | dest

1212 ROP_N_OR_N: pixel_color = ~src | ~dest

1213 ROP_XOR: pixel_color = src ^ dest

1214 ROP_EOR: pixel_color = src ^ ~dest

1215 **Sub-elements:** N/A

1216 4.12.2.13 TEXT_DIR

1217 **Semantics:** Set the current text direction. The direction specifies that line along which successive character
1218 origin points are placed (see figure 4); that is the line from one glyph origin to the next glyph origin.

1219 **Properties:**

1220 $v1$: a character string, representing the text direction. The possible values for this property are
1221 HEAD_LEFT, HEAD_RIGHT, HEAD_TOP and HEAD_BOTTOM. HEAD_LEFT is the text direction is from left
1222 to right. HEAD_RIGHT is the text direction is from right to left. HEAD_TOP is the text direction is from
1223 top to bottom. HEAD_BOTTOM is the text direction is from bottom to top.

1224 **Sub-elements:** N/A

1225 4.12.2.14 CHAR_DIR

1226 **Semantics:** Set the current character direction (e.g., the direction in which a character is rendered). The
1227 heading direction is from the bottom of a character to the top.

1228 **Properties:**

1229 $v1$: a character string representing the character direction. The possible values for this property are
1230 HEAD_LEFT, HEAD_RIGHT, HEAD_TOP and HEAD_BOTTOM. HEAD_LEFT is the character's heading
1231 direction is left. HEAD_RIGHT is the character's heading direction is right. HEAD_TOP is the character's
1232 heading direction is up. HEAD_BOTTOM is the character's heading direction is down.

1233 **Sub-elements:** N/A

1234 4.12.2.15 CHAR_ROTATE

1235 **Semantics:** Set the current character rotation angle.

1236 **Properties:**

1237 $v1$: a floating point number, representing the character rotating radian. A positive value represents
1238 counterclockwise; a negative value represents clockwise.

1239 $v2$: a character string, representing whether the rotation is around the character center or around the
1240 top-left corner. The possible values for this property are ROT_CENTER and ROT_LEFTTOP.

1241 **Sub-elements:** N/A

1242 4.12.2.16 CHAR_SLANT

1243 **Semantics:** Set the slant of the character.

1244 **Properties:**

1245 $v1$: a floating point number, representing the character slanting radian, regardless of reading direction.
1246 $0 \sim \pi/2$ represents right slant, $3\pi/2 \sim 2\pi$ represents left slant, and 0 represents non-slant; other values
1247 are not used.

1248 **Sub-elements:** N/A

1249 4.12.2.17 CHAR_SIZE

1250 **Semantics:** Set the current character width and height.

1251 **Properties:**

1252 $v1$: a positive floating point number, representing the character width.

1253 $v2$: a positive floating point number, representing the character height.

1254 **Sub-elements:** N/A

1255 4.12.2.18 CHAR_WEIGHT

1256 **Semantics:** Set the current character weight. The default value is 0. The thickness of a character stroke shall be
1257 the normal thickness plus $\text{weight} * (\text{character height})$. The minimum thickness of a character's stroke is zero.

1258 **Properties:**

1259 *v1*: a floating point number, ranging between -1 to 1, inclusively, representing the character weight.

1260 **Sub-elements:** N/A

1261 4.12.2.19 CHAR_STYLE

1262 **Semantics:** Set the current character style.

1263 **Properties:**

1264 *v1*: a character string, representing the character style. The possible values for this property are
1265 SHADOW, HOLLOW and OUTLINE, or some combination of the three, separated by commas. If the
1266 string is set to empty, then any previous setting is cleared.

1267

1268 SHADOW: set shadow style. If this character style is set, then the following algorithm is used to render
1269 the shadow effect:

1270

- 1271 • If SHADOW_NEG (§4.11.2.30) is false, the character is extended with a distance of
1272 SHADOW_LEN (§4.11.2.27) along the shadow direction (§4.11.2.28), then a hollowed character
1273 with raster operation ROP_COPY is drawn in the original position. The border width of the
1274 hollowed character is SHADOW_WIDTH (§4.11.2.26).
- 1275 • If SHADOW_NEG is true, the character position is moved with a distance of SHADOW_LEN
1276 along the shadow direction, and extended SHADOW_WIDTH along the shadow direction; then
1277 the character is drawn in the original position with background color and raster operation
1278 ROP_COPY, and extended with a distance SHADOW_LEN along the shadow direction; then in
1279 the original position, a character with normal color and raster operation ROP_COPY is drawn.
1280

1281



1282

1283 HOLLOW: set hollow style. If this character style is set, a line with thickness HOLLOW_BORDER
1284 (§4.11.2.35) should be drawn along the outline of the character.



1285

1286

1287

1288

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.

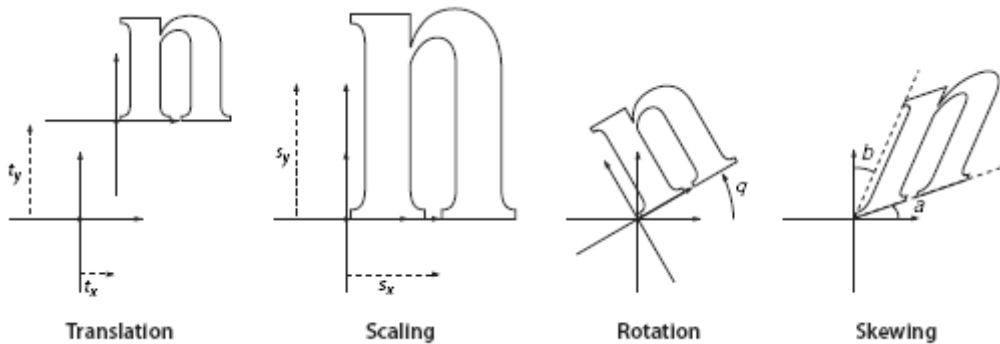
1289

1290 **Sub-elements:** N/A

1291

1292 4.12.2.20 `TEXT_MATRIX`

1293 **Semantics:** Set the current text transformation matrix. This command applies to each character individually
 1294 within a `TEXT` object. The visual effect of transforming a character is shown below:



1295

1296 **Properties:** N/A

1297 **Sub-elements:**

1298 *matrix*: element of the `MATRIX` (§4.11.3.2) type, responsible for transforming coordinates of text.

1299 4.12.2.21 `IMAGE_MATRIX`

1300 **Semantics:** Set the current image transformation matrix

1301 **Properties:** N/A

1302 **Sub-elements:**

1303 *matrix*: element of `MATRIX` (§4.12.3.2) type, used for transforming coordinates of an image.

1304 4.12.2.22 `GRAPH_MATRIX`

1305 **Semantics:** Set the current line/curve transformation matrix

1306 **Properties:** N/A

1307 **Sub-elements:**

1308 *matrix*: element of the MATRIX (§4.11.3.2) type, used for transforming the coordinates of path
1309 graphics, such as line, Bezier curve, arc, circle, ellipse, rect, roundrect, subpath, path, etc.

1310 4.12.2.23 EXT_MATRIX

1311 **Semantics:** Set the current extension transformation matrix

1312 **Properties:** N/A

1313 **Sub-elements:**

1314 *matrix*: element of the MATRIX (§4.11.3.2) type, used for transforming the coordinates of all path
1315 graphics, images and texts. The current extension transformation matrix is applied to the object after
1316 any current dedicated transformation matrix has been applied to the object.

1317 4.12.2.24 PUSH_GS

1318 **Semantics:** Push the current graphics state onto the graphics state stack.

1319 **Properties:** N/A

1320 **Sub-elements:** N/A

1321 4.12.2.25 POP_GS

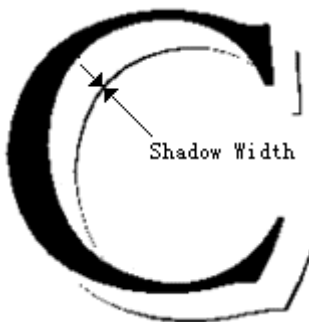
1322 **Semantics:** Pop out the top value from the graphics state stack, replacing the current graphics state.

1323 **Properties:** N/A

1324 **Sub-elements:** N/A

1325 4.12.2.26 SHADOW_WIDTH

1326 **Semantics:** Set the border width of the current character shadow. SHADOW_WIDTH represents the thickness of
1327 the outline of a shadow.



1328

1329 **Properties:**

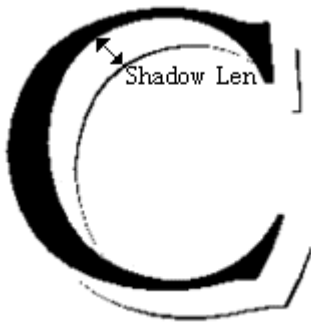
1330 *v1*: a non-negative floating point number, representing the shadow border width.

1331 **Sub-elements:** N/A

1332

1333 4.12.2.27 SHADOW_LEN

1334 **Semantics:** Set the length of the current character shadow. SHADOW_LEN represents the displacement of the
1335 shadow with respect to the character.



1336

1337 **Properties:**

1338 `v1`: a non-negative floating point number, representing the character shadow length.

1339 **Sub-elements:** N/A

1340

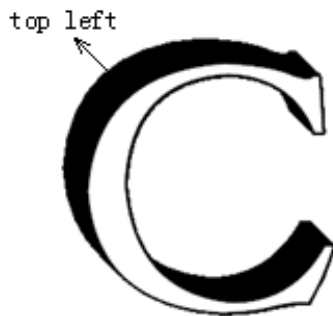
1341 **4.12.2.28 SHADOW_DIR**

1342 **Semantics:** Set the direction of the current character shadow

1343 **Properties:**

1344 `v1`: a character string. The possible values for this property are SHADOW_LT, SHADOW_LB,
1345 SHADOW_RT and SHADOW_RB. Choosing one of these values specifies which direction the character
1346 shadow will be seen.

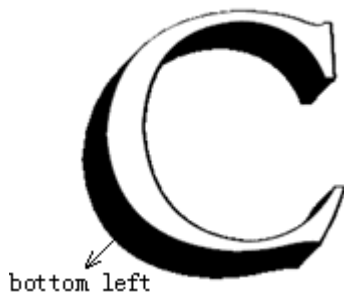
1347 SHADOW_LT: the character shadow direction is top left.



1348

1349

1350 SHADOW_LB: the character shadow direction is bottom left.



1351

1352

1353 SHADOW_RT: the character shadow direction is top right.

1354



1355

1356

SHADOW_RB: the character shadow direction is bottom right.

1357



1358

1359

1360 **Sub-elements:** N/A

1361

1362 4.12.2.29 SHADOW_ATL

1363 **Semantics:** Set whether to adjust the coordinates of a character when the direction of character shadow is to
1364 the left or bottom.

1365 **Properties:**

1366 **v1:** a Boolean value, representing whether to alter the coordinates of a character. The value 'true'
1367 specifies that the coordinates are altered.

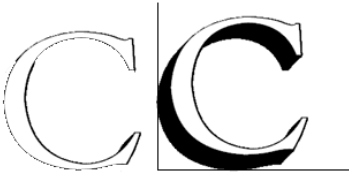
1368 **Sub-elements:** N/A

1369 [Example: Illustrated in the figures below, when a character is shadowed, the bounding box of its outline is
1370 bigger. If two characters that are not shadowed are adjacent, their baselines are aligned horizontally. A shadow
1371 effect will break this horizontal alignment. Also, a shadow to the left will occupy the space between this
1372 character and its left neighbor. When a rendering engine draws the character, it can position the character
1373 based on the specific coordinate; or it can adjust the coordinate so that the bottom left point of the shadowed
1374 character's outline bounding box moves to the specific coordinate. This is made by offset x or y coordinates by
1375 the distance of SHADOW_LEN divided by the square root of 2. When the shadow is to the bottom of the
1376 character, subtract y by the distance; when the shadow is to the left, add x by the distance. Make both
1377 adjustments when the shadow is to the bottom left. This explains the parameter SHADOW_ATL. When
1378 SHADOW_ATL is false, the specific coordinate is used without adjustment; when it is true, an adjustment
1379 should be made. The first figure illustrates the effect before adjustment, while the second figure illustrates the

1380 effect after adjustment.



1381



1382

1383

1384

1385 *end example]*

1386 4.12.2.30 SHADOW_NEG

1387 **Semantics:** Set the current shadow character as an intaglio character as illustrated in the following figures.



1388

1389 SHADOW_NEG is false



1390

1391 SHADOW_NEG is true

1392

1393 **Properties:**

1394 *v1*: a boolean value, representing whether the current shadow character is an intaglio character. A
1395 'true' value specifies an intaglio character.

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

1397 **4.12.2.31 CLIP_AREA**

1398 **Semantics:** Set the current clip area

1399 **Properties:** N/A

1400 **Sub-elements:**

1401 *cliparea*: PATH type, representing the new clip area.

1402 The Path specified by a CLIP_AREA command object is relative to the page. The portions of graphic
1403 objects that lie outside of the current clip area are not rendered.

1404 **4.12.2.32 FONT**

1405 **Semantics:** set the font used by an encoding/character set. [*Example*: set an English character to use the font
1406 named "Arial". *end example*]

1407

1408 **Properties:**

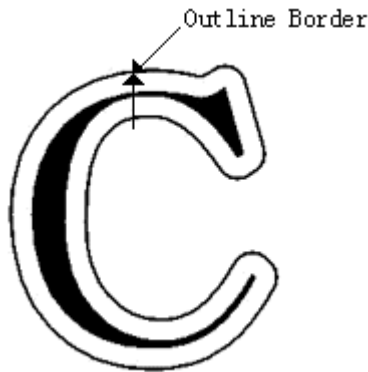
1409 *v1*: a character string, representing the encoding/character set. The valid value for this property is the
1410 same as for the *encode* property of TEXT (§4.10.11).

1411 *v2*: a character string, representing the font that will be used by the encoding/character set.

1412 **Sub-elements:** N/A

1413 **4.12.2.33 OUTLINE_BORDER**

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



1415

1416 **Properties:**

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

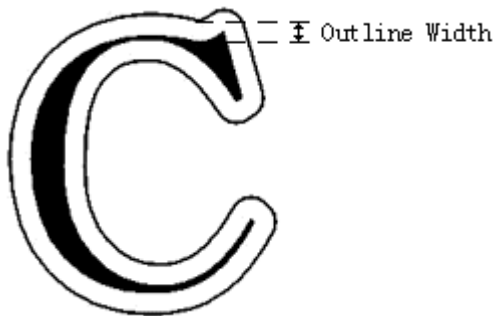
1418 **Sub-elements:** N/A

1419

1420

1421 **4.12.2.34** **OUTLINE_WIDTH**

1422 **Semantics:** Set the outline width of the current outline character



1423

1424 **Properties:**

1425 $v1$: a non-negative floating point number, representing the outline width.

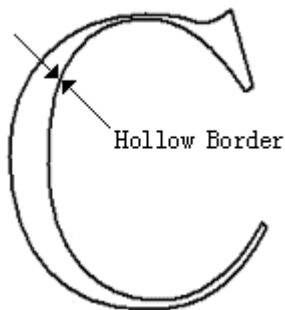
1426 **Sub-elements:** N/A

1427

1428

1429 **4.12.2.35** **HOLLOW_BORDER**

1430 **Semantics:** Set the border width of the current hollow character



1431

1432 **Properties:**

1433 $v1$: a non-negative floating point number, representing the border width.

1434 **Sub-elements:** N/A

1435

1436 **4.12.3** **Definition of Referenced Type**

1437 This clause specifies the definition of the data types referred in the UOML XML schema descriptions.

1438 **4.12.3.1 COLOR_RGB**

1439 **Semantics:** the value of a color setting

1440 **Properties:**

1441 *r*: red component

1442 *g*: green component

1443 *b*: blue component

1444 *a*: optional alpha component.

1445 **Sub-element:** N/A

1446 **4.12.3.2 MATRIX**

1447 **Semantics:** the values in a transformation matrix

1448 **Properties:**

1449 *f11*: floating point number

1450 *f12*: floating point number

1451 *f21*: floating point number

1452 *f22*: floating point number

1453 *f31*: floating point number

1454 *f32*: floating point number

1455 **Sub-element:** N/A

1456 [*Note:*

1457 A transformation of matrix in UOML is specified by six numbers. In an abbreviated notation, this array
1458 is denoted [*f11 f12 f21 f22 f31 f32*]; it can represent any linear transformation from one coordinate
1459 system to another. The transformation is carried out as follows:

1460

1461 $x' = f11 \times x + f21 \times y + f31$

1462 $y' = f12 \times x + f22 \times y + f32$

1463

1464 • Translations are specified using [*1 0 0 1 tx ty*], where *tx* and *ty* shall be the distances to translate the
1465 origin of the coordinate system in the horizontal and vertical dimensions, respectively.

1466 • Scaling is specified using [*sx 0 0 sy 0 0*]. This scales the coordinates so that 1 unit in the horizontal
1467 and vertical dimensions of the new coordinate system is the same size as *sx* and *sy* units, respectively,
1468 in the previous coordinate system.

1469 • Rotations are specified using by [*cos(q) sin(q) -sin(q) cos(q) 0 0*], which has the effect of rotating the
1470 coordinate system axes by an angle *q* counterclockwise.

1471 • Skew is specified using [*1 tan(a) tan(b) 1 0 0*], which skews the *x* axis by an angle *a* and the *y* axis by
1472 an angle *b*.

1473

1474 *end note*]

1475 **4.13 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

1476

1477

1478

4.14 Definition of Parameter Data Types

1479

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

1480 **4.14.1 INT**

1481 **Properties:**

1482 *name*: a character string value, xs:string type

1483 *val*: xs:integer type

1484 **Sub-element:** N/A

1485 **4.14.2 DOUBLE**

1486 **Properties:**

1487 *name*: a character string, xs:string type

1488 *val*: xs:double type

1489 **Sub-element:** N/A

1490 **4.14.3 LONG**

1491 **Properties:**

1492 *name*: a character string, xs:string type

1493 *val*: xs:long type

1494 **Sub-element:** N/A

1495 **4.14.4 DATE**

1496 **Properties:**

1497 *name*: a character string, xs:string type

1498 *val*: xs:date type

1499 **Sub-element:** N/A

1500 **4.14.5 TIME**

1501 **Properties:**

1502 *name*: a character string, xs:string type

1503 *val*: xs:time type

1504 **Sub-element:** N/A

1505 **4.14.6 DATETIME**

1506 **Properties:**

1507 *name*: a character string, xs:string type

1508 *val*: xs:datetime type

1509 **Sub-element:** N/A

1510 **4.14.7 DURATION**

1511 **Properties:**

1512 *name*: a character string, xs:string type

1513 *val*: xs:duration type

1514 **Sub-element:** N/A

1515 **4.14.8 STRING**

1516 **Properties:**

1517 *name*: a character string, xs:string type

1518 *val*: xs:string type

1519 **Sub-element:** N/A

1520 **4.14.9 BINARY**

1521 **Properties:**

1522 *name*: a character string, xs:string type

1523 *val*: xs:base64Binary type

1524 **Sub-element:** N/A

1525 **4.14.10 BOOL**

1526 **Properties:**

1527 *name*: a character string, xs:string type

1528 *val*: xs:boolean type

1529 **Sub-element:** N/A

1530 **4.14.11 COMPOUND**

1531 **Property:**

1532 *name*: a character string, xs:string type

1533 **Sub-element:**

1534 *arc*: ARC type

1535 *bezier*: BEZIER type

1536 *circle*: CIRCLE type

1537 *cmd*: CMD type

1538 *rgb*: COLOR_RGB type

1539 *doc*: DOC type

1540 *docbase*: DOCBASE type

1541 *docset*: DOCSET type

1542 *ellipse*: ELLIPSE type

1543 *embedfont*: EMBEDFONT type
1544 *fontlist*: FONTLIST type
1545 *fontmap*: FONTMAP type
1546 *image*: IMAGE type
1547 *layer*: LAYER type
1548 *line*: LINE type
1549 *matrix*: MATRIX type
1550 *meta*: META type
1551 *metalist*: METALIST type
1552 *page*: PAGE type
1553 *path*: PATH type
1554 *rect*: RECT type
1555 *roundrect*: ROUNDRECT type
1556 *subpath*: SUBPATH type
1557 *text*: TEXT type
1558 *objstream*: OBJSTREAM type

1559 [Note: Each sub-element may occur zero or more times. *end note*]

1560

1561 4.15 Data Ranges

1562 The following are the general rules for data ranges:

1563

- 1564 1. Unless otherwise specified, all numeric values may be positive, negative or zero.
- 1565 2. Positive, negative, or zero values are allowed for coordinates and points in the logical coordinate
1566 system (e.g. -1, 3).
- 1567 3. Integer values are 32-bit precision; the range of integer values is as defined by xs:integer in XML
1568 Schema 1.0 Part 2.
- 1569 4. Float values use double-precision; the valid range is as defined by xs:double in XML Schema 1.0
1570 Part 2.
- 1571 5. API calls that set values outside a valid range (either specifically specified or within the ranges
1572 above) will fail with a return of RET.
- 1573 6. A special case is COLOR_RGB. RGB32 is used, thus each property of COLOR_RGB(r, g, b, a) falls
1574 within a range of 0-255.
- 1575 7. Valid ranges and formats for a date are as defined by xs:date in XML Schema 1.0 Part 2.

1576

1577

5. Conformance

1578 The text in this OASIS standard is divided into *normative* and *informative* categories. Unless documented
1579 otherwise, all features specified in normative text of this OASIS standard shall be implemented. Text marked
1580 informative (using the mechanisms described in §1.5) is for information purposes only. Unless stated
1581 otherwise, all text is normative.

1582 Use of the word “shall” indicates required behavior.

1583 Any behavior that is not explicitly specified by this OASIS standard is implicitly unspecified (§4).

1584 5.1.1 DCMS Conformance

1585 An UOML Document Management System (DCMS) has conformance if it implements all of the UOML
1586 instructions in compliance with the syntax as described in the schema [UOMLSchema] and semantics in this
1587 OASIS standard.

1588 5.1.2 Application Conformance

1589 An UOML application is conformant if both of the following are true:

- 1590 • The application issues UOML instructions as schema-valid XML] as specified in this OASIS standard to
1591 the DCMS; and
- 1592 • The application parses the return instructions from the DCMS according to this OASIS standard.

1593

Annex A.UOML XML Schema

1594 **This annex is informative.**

1595 The following is a copy of the XML Schema for UOML for ancillary purposes. It describes the types and
1596 elements, in XML format, for UOML. The normative schema is provided with the specification.

1597 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..](http://docs.oasis-open.org/uoml-x/v1.0/errata/cd/uoml-
1598 part1-v1.0-schema-errata.xsd..)

```
1599 <?xml version="1.0" encoding="UTF-8"?>
1600 <xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
1601 xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0"
1602 targetNamespace="urn:oasis:names:tc:uoml:xmlns:uoml:1.0"
1603 elementFormDefault="unqualified" attributeFormDefault="unqualified">
1604     <xs:complexType name="ARC">
1605         <xs:annotation>
1606             <xs:documentation>arc</xs:documentation>
1607         </xs:annotation>
1608         <xs:attribute name="clockwise" type="xs:boolean" use="required"/>
1609         <xs:attribute name="start" type="xs:string" use="required"/>
1610         <xs:attribute name="end" type="xs:string" use="required"/>
1611         <xs:attribute name="center" type="xs:string" use="required"/>
1612         <xs:attribute name="angle" type="xs:float" use="required"/>
1613     </xs:complexType>
1614     <xs:complexType name="BEZIER">
1615         <xs:annotation>
1616             <xs:documentation>bezier curve</xs:documentation>
1617         </xs:annotation>
1618         <xs:attribute name="start" type="xs:string" use="required"/>
1619         <xs:attribute name="ctrl" type="xs:string" use="required"/>
1620         <xs:attribute name="ctrl2" type="xs:string" use="optional"/>
1621         <xs:attribute name="end" type="xs:string" use="required"/>
1622     </xs:complexType>
1623     <xs:complexType name="CIRCLE">
1624         <xs:annotation>
1625             <xs:documentation>circle</xs:documentation>
1626         </xs:annotation>
1627         <xs:attribute name="radius" type="xs:int" use="required"/>
1628         <xs:attribute name="center" type="xs:string" use="required"/>
1629     </xs:complexType>
1630     <xs:complexType name="LINE">
1631         <xs:annotation>
1632             <xs:documentation>line</xs:documentation>
1633         </xs:annotation>
```



```

1634         <xs:attribute name="start" type="xs:string" use="required"/>
1635         <xs:attribute name="end" type="xs:string" use="required"/>
1636     </xs:complexType>
1637     <xs:complexType name="RECT">
1638         <xs:annotation>
1639             <xs:documentation>rect</xs:documentation>
1640         </xs:annotation>
1641         <xs:attribute name="tl" type="xs:string" use="required"/>
1642         <xs:attribute name="br" type="xs:string" use="required"/>
1643     </xs:complexType>
1644     <xs:complexType name="ELLIPSE">
1645         <xs:annotation>
1646             <xs:documentation>ellipse</xs:documentation>
1647         </xs:annotation>
1648         <xs:attribute name="xr" type="xs:int" use="required"/>
1649         <xs:attribute name="yr" type="xs:int" use="required"/>
1650         <xs:attribute name="center" type="xs:string" use="required"/>
1651         <xs:attribute name="angle" type="xs:float" use="required"/>
1652     </xs:complexType>
1653     <xs:complexType name="ROUNDRECT">
1654         <xs:annotation>
1655             <xs:documentation>roundrect</xs:documentation>
1656         </xs:annotation>
1657         <xs:attribute name="xr" type="xs:int" use="required"/>
1658         <xs:attribute name="yr" type="xs:int" use="required"/>
1659         <xs:attribute name="tl" type="xs:string" use="required"/>
1660         <xs:attribute name="br" type="xs:string" use="required"/>
1661     </xs:complexType>
1662     <xs:complexType name="META">
1663         <xs:annotation>
1664             <xs:documentation>metadata</xs:documentation>
1665         </xs:annotation>
1666         <xs:attribute name="key" type="xs:string" use="required"/>
1667         <xs:attribute name="val" type="xs:string" use="required"/>
1668     </xs:complexType>
1669     <xs:complexType name="METALIST">
1670         <xs:annotation>
1671             <xs:documentation>metadata list</xs:documentation>
1672         </xs:annotation>
1673         <xs:sequence>
1674             <xs:element name="meta" type="uoml:META" minOccurs="0"
1675 maxOccurs="unbounded"/>
1676         </xs:sequence>
1677     </xs:complexType>
1678     <xs:complexType name="CMD">
1679         <xs:annotation>
1680             <xs:documentation>cmd</xs:documentation>
1681         </xs:annotation>
1682         <xs:sequence minOccurs="0">

```

```

1683         <xs:choice>
1684             <xs:element name="cliparea" type="uoml:PATH"/>
1685             <xs:element name="matrix" type="uoml:MATRIX"/>
1686             <xs:element name="rgb" type="uoml:COLOR_RGB"/>
1687         </xs:choice>
1688     </xs:sequence>
1689     <xs:attribute name="name" type="uoml:CMDNAME" use="required"/>
1690     <xs:attribute name="v1" type="xs:anySimpleType"/>
1691     <xs:attribute name="v2" type="xs:anySimpleType"/>
1692 </xs:complexType>
1693 <xs:complexType name="MATRIX">
1694     <xs:annotation>
1695         <xs:documentation>matrix</xs:documentation>
1696     </xs:annotation>
1697     <xs:attribute name="f11" type="xs:float" use="required"/>
1698     <xs:attribute name="f12" type="xs:float" use="required"/>
1699     <xs:attribute name="f21" type="xs:float" use="required"/>
1700     <xs:attribute name="f22" type="xs:float" use="required"/>
1701     <xs:attribute name="f31" type="xs:float" use="required"/>
1702     <xs:attribute name="f32" type="xs:float" use="required"/>
1703 </xs:complexType>
1704 <xs:complexType name="SUBPATH">
1705     <xs:annotation>
1706         <xs:documentation>subpath</xs:documentation>
1707     </xs:annotation>
1708     <xs:attribute name="data" type="xs:string" use="required"/>
1709 </xs:complexType>
1710 <xs:complexType name="PATH">
1711     <xs:annotation>
1712         <xs:documentation>path</xs:documentation>
1713     </xs:annotation>
1714     <xs:sequence>
1715         <xs:choice minOccurs="0" maxOccurs="unbounded">
1716             <xs:element name="subpath" type="uoml:SUBPATH"/>
1717             <xs:element name="rect" type="uoml:RECT"/>
1718             <xs:element name="circle" type="uoml:CIRCLE"/>
1719             <xs:element name="ellipse" type="uoml:ELLIPSE"/>
1720             <xs:element name="roundrect" type="uoml:ROUNDRECT"/>
1721         </xs:choice>
1722     </xs:sequence>
1723 </xs:complexType>
1724 <xs:complexType name="COLOR_RGB">
1725     <xs:annotation>
1726         <xs:documentation>rgb color</xs:documentation>
1727     </xs:annotation>
1728     <xs:attribute name="r" type="xs:short" use="required"/>
1729     <xs:attribute name="g" type="xs:short" use="required"/>
1730     <xs:attribute name="b" type="xs:short" use="required"/>
1731     <xs:attribute name="a" type="xs:short" use="optional"/>

```

```

1732     </xs:complexType>
1733     <xs:complexType name="EMBEDFONT">
1734         <xs:annotation>
1735             <xs:documentation>embedded font</xs:documentation>
1736         </xs:annotation>
1737         <xs:simpleContent>
1738             <xs:extension base="xs:base64Binary">
1739
1740                 </xs:extension>
1741             </xs:simpleContent>
1742         </xs:complexType>
1743     <xs:complexType name="FONTMAP">
1744         <xs:annotation>
1745             <xs:documentation>font mapping</xs:documentation>
1746         </xs:annotation>
1747         <xs:attribute name="name" type="xs:string" use="required"/>
1748         <xs:attribute name="no" type="xs:int" use="required"/>
1749     </xs:complexType>
1750     <xs:complexType name="FONTLIST">
1751         <xs:annotation>
1752             <xs:documentation>font list</xs:documentation>
1753         </xs:annotation>
1754     </xs:complexType>
1755     <xs:complexType name="IMAGE">
1756         <xs:annotation>
1757             <xs:documentation>image</xs:documentation>
1758         </xs:annotation>
1759         <xs:simpleContent>
1760             <xs:extension base="xs:base64Binary">
1761                 <xs:attribute name="tl" type="xs:string" use="required"/>
1762                 <xs:attribute name="br" type="xs:string" use="required"/>
1763                 <xs:attribute name="type" type="xs:string" use="required"/>
1764                 <xs:attribute name="path" type="xs:string" use="optional"/>
1765             </xs:extension>
1766         </xs:simpleContent>
1767
1768     </xs:complexType>
1769     <xs:complexType name="TEXT">
1770         <xs:annotation>
1771             <xs:documentation>text</xs:documentation>
1772         </xs:annotation>
1773         <xs:attribute name="origin" type="xs:string" use="required"/>
1774         <xs:attribute name="encode" type="xs:string" use="required"/>
1775         <xs:attribute name="text" type="xs:string" use="required"/>
1776         <xs:attribute name="spaces" type="xs:string" use="optional"/>
1777     </xs:complexType>
1778     <xs:simpleType name="CMDNAME">
1779         <xs:annotation>
1780             <xs:documentation>command names</xs:documentation>
1781         </xs:annotation>

```

```

1780     <xs:restriction base="xs:string">
1781         <xs:enumeration value="COLOR_LINE"/>
1782         <xs:enumeration value="COLOR_FILL"/>
1783         <xs:enumeration value="COLOR_TEXT"/>
1784         <xs:enumeration value="COLOR_SHADOW"/>
1785         <xs:enumeration value="COLOR_OUTLINE"/>
1786         <xs:enumeration value="LINE_WIDTH"/>
1787         <xs:enumeration value="LINE_JOIN"/>
1788         <xs:enumeration value="LINE_CAP"/>
1789         <xs:enumeration value="MITER_LIMIT"/>
1790         <xs:enumeration value="FILL_RULE"/>
1791         <xs:enumeration value="RENDER_MODE"/>
1792         <xs:enumeration value="RASTER_OP"/>
1793         <xs:enumeration value="TEXT_DIR"/>
1794         <xs:enumeration value="CHAR_DIR"/>
1795         <xs:enumeration value="CHAR_ROTATE"/>
1796         <xs:enumeration value="CHAR_SLANT"/>
1797         <xs:enumeration value="CHAR_SIZE"/>
1798         <xs:enumeration value="CHAR_WEIGHT"/>
1799         <xs:enumeration value="CHAR_STYLE"/>
1800         <xs:enumeration value="TEXT_MATRIX"/>
1801         <xs:enumeration value="IMAGE_MATRIX"/>
1802         <xs:enumeration value="GRAPH_MATRIX"/>
1803         <xs:enumeration value="EXT_MATRIX"/>
1804         <xs:enumeration value="PUSH_GS"/>
1805         <xs:enumeration value="POP_GS"/>
1806         <xs:enumeration value="SHADOW_WIDTH"/>
1807         <xs:enumeration value="SHADOW_DIR"/>
1808         <xs:enumeration value="SHADOW_LEN"/>
1809         <xs:enumeration value="SHADOW_NEG"/>
1810         <xs:enumeration value="SHADOW_ATL"/>
1811         <xs:enumeration value="CLIP_AREA"/>
1812         <xs:enumeration value="FONT"/>
1813         <xs:enumeration value="OUTLINE_BORDER"/>
1814         <xs:enumeration value="OUTLINE_WIDTH"/>
1815         <xs:enumeration value="HOLLOW_BORDER"/>
1816     </xs:restriction>
1817 </xs:simpleType>
1818 <xs:simpleType name="LINECAP">
1819     <xs:annotation>
1820         <xs:documentation>line cap style</xs:documentation>
1821     </xs:annotation>
1822     <xs:restriction base="xs:string">
1823         <xs:enumeration value="END_BUTT"/>
1824         <xs:enumeration value="END_SQUARE"/>
1825         <xs:enumeration value="END_ROUND"/>
1826     </xs:restriction>
1827 </xs:simpleType>
1828 <xs:simpleType name="JOINCAP">

```

```

1829     <xs:annotation>
1830         <xs:documentation>line join style</xs:documentation>
1831     </xs:annotation>
1832     <xs:restriction base="xs:string">
1833         <xs:enumeration value="JOIN_MITER"/>
1834         <xs:enumeration value="JOIN_BEVEL"/>
1835         <xs:enumeration value="JOIN_ROUND"/>
1836     </xs:restriction>
1837 </xs:simpleType>
1838 <xs:simpleType name="FILLRULE">
1839     <xs:annotation>
1840         <xs:documentation>fill rule</xs:documentation>
1841     </xs:annotation>
1842     <xs:restriction base="xs:string">
1843         <xs:enumeration value="RULE_EVENODD"/>
1844         <xs:enumeration value="RULE_WINDING"/>
1845     </xs:restriction>
1846 </xs:simpleType>
1847 <xs:simpleType name="ROP">
1848     <xs:annotation>
1849         <xs:documentation>rop operation</xs:documentation>
1850     </xs:annotation>
1851     <xs:restriction base="xs:string">
1852         <xs:enumeration value="ROP_COPY"/>
1853         <xs:enumeration value="ROP_N_COPY"/>
1854         <xs:enumeration value="ROP_RESET"/>
1855         <xs:enumeration value="ROP_SET"/>
1856         <xs:enumeration value="ROP_NOP"/>
1857         <xs:enumeration value="ROP_REV"/>
1858         <xs:enumeration value="ROP_AND"/>
1859         <xs:enumeration value="ROP_AND_N"/>
1860         <xs:enumeration value="ROP_N_AND"/>
1861         <xs:enumeration value="ROP_N_AND_N"/>
1862         <xs:enumeration value="ROP_OR"/>
1863         <xs:enumeration value="ROP_OR_N"/>
1864         <xs:enumeration value="ROP_N_OR"/>
1865         <xs:enumeration value="ROP_N_OR_N"/>
1866         <xs:enumeration value="ROP_XOR"/>
1867         <xs:enumeration value="ROP_EOR"/>
1868     </xs:restriction>
1869 </xs:simpleType>
1870 <xs:simpleType name="CHARTXTDIR">
1871     <xs:annotation>
1872         <xs:documentation>text or char direction</xs:documentation>
1873     </xs:annotation>
1874     <xs:restriction base="xs:string">
1875         <xs:enumeration value="HEAD_LEFT"/>
1876         <xs:enumeration value="HEAD_RIGHT"/>
1877         <xs:enumeration value="HEAD_TOP"/>

```

```

1878         <xs:enumeration value="HEAD_BOTTOM"/>
1879     </xs:restriction>
1880 </xs:simpleType>
1881 <xs:simpleType name="SHADOWDIR">
1882     <xs:annotation>
1883         <xs:documentation>shadow direction</xs:documentation>
1884     </xs:annotation>
1885     <xs:restriction base="xs:string">
1886         <xs:enumeration value="SHADOW_LT"/>
1887         <xs:enumeration value="SHADOW_LB"/>
1888         <xs:enumeration value="SHADOW_RT"/>
1889         <xs:enumeration value="SHADOW_RB"/>
1890     </xs:restriction>
1891 </xs:simpleType>
1892 <xs:complexType name="OBJSTREAM">
1893     <xs:annotation>
1894         <xs:documentation>object stream</xs:documentation>
1895     </xs:annotation>
1896 </xs:complexType>
1897 <xs:complexType name="LAYER">
1898     <xs:annotation>
1899         <xs:documentation>layer</xs:documentation>
1900     </xs:annotation>
1901 </xs:complexType>
1902 <xs:complexType name="PAGE">
1903     <xs:annotation>
1904         <xs:documentation>page</xs:documentation>
1905     </xs:annotation>
1906     <xs:attribute name="width" type="xs:float" use="required"/>
1907     <xs:attribute name="height" type="xs:float" use="required"/>
1908     <xs:attribute name="resolution" type="xs:int" use="required"/>
1909 </xs:complexType>
1910 <xs:complexType name="DOC">
1911     <xs:annotation>
1912         <xs:documentation>doc</xs:documentation>
1913     </xs:annotation>
1914     <xs:sequence>
1915         <xs:element name="metainfo" type="uoml:METALIST"/>
1916     </xs:sequence>
1917     <xs:attribute name="name" type="xs:string" use="required"/>
1918 </xs:complexType>
1919 <xs:complexType name="DOCSET">
1920     <xs:annotation>
1921         <xs:documentation>doc set</xs:documentation>
1922     </xs:annotation>
1923     <xs:attribute name="name" type="xs:string" use="required"/>
1924 </xs:complexType>
1925 <xs:complexType name="DOCBASE">
1926     <xs:annotation>

```

```

1927         <xs:documentation>doc base</xs:documentation>
1928     </xs:annotation>
1929     <xs:attribute name="name" type="xs:string" use="required"/>
1930     <xs:attribute name="path" type="xs:string" use="required"/>
1931 </xs:complexType>
1932 <xs:element name="CLOSE">
1933     <xs:complexType>
1934         <xs:attribute name="handle" type="xs:string" use="optional"/>
1935     </xs:complexType>
1936 </xs:element>
1937 <xs:element name="DELETE">
1938     <xs:complexType>
1939         <xs:attribute name="handle" type="xs:string" use="optional"/>
1940     </xs:complexType>
1941 </xs:element>
1942 <xs:element name="INSERT">
1943     <xs:complexType>
1944         <xs:choice>
1945             <xs:element name="xobj" type="uoml:COMPOUND"/>
1946         </xs:choice>
1947         <xs:attribute name="handle" type="xs:string"/>
1948         <xs:attribute name="pos" type="xs:int"/>
1949     </xs:complexType>
1950 </xs:element>
1951 <xs:element name="GET">
1952     <xs:complexType>
1953         <xs:choice>
1954             <xs:element name="disp_conf">
1955                 <xs:complexType>
1956                     <xs:sequence>
1957                         <xs:element name="clip" type="uoml:PATH"
1958 minOccurs="0"/>
1959                     </xs:sequence>
1960                     <xs:attribute name="end_layer" type="xs:int"/>
1961                     <xs:attribute name="resolution"
1962 type="xs:int"/>
1963                     <xs:attribute name="format" type="xs:string"/>
1964                     <xs:attribute name="output" type="xs:string"
1965 use="required"/>
1966                     <xs:attribute name="addr" type="xs:string"
1967 use="required"/>
1968                 </xs:complexType>
1969             </xs:element>
1970             <xs:element name="pos">
1971                 <xs:complexType>
1972                     <xs:attribute name="val" type="xs:int"
1973 use="required"/>
1974                 </xs:complexType>
1975             </xs:element>

```

```

1976         <xs:element name="property">
1977             <xs:complexType>
1978                 <xs:attribute name="name" type="xs:string"
1979 use="required"/>
1980             </xs:complexType>
1981         </xs:element>
1982     </xs:choice>
1983     <xs:attribute name="usage" type="xs:string" use="required"/>
1984     <xs:attribute name="handle" type="xs:string"/>
1985 </xs:complexType>
1986 </xs:element>
1987 <xs:element name="SET">
1988     <xs:complexType>
1989     <xs:choice>
1990         <xs:choice minOccurs="0" maxOccurs="unbounded">
1991             <xs:element name="intVal" type="uoml:INT"/>
1992             <xs:element name="floatVal" type="uoml:DOUBLE"/>
1993             <xs:element name="timeVal" type="uoml:TIME"/>
1994             <xs:element name="dateVal" type="uoml:DATE"/>
1995             <xs:element name="dateTimeVal"
1996 type="uoml:DATETIME"/>
1997             <xs:element name="durationVal"
1998 type="uoml:DURATION"/>
1999             <xs:element name="stringVal" type="uoml:STRING"/>
2000             <xs:element name="binaryVal" type="uoml:BINARY"/>
2001             <xs:element name="compoundVal"
2002 type="uoml:COMPOUND"/>
2003             <xs:element name="boolVal" type="uoml:BOOL"/>
2004         </xs:choice>
2005     </xs:choice>
2006     <xs:attribute name="handle" type="xs:string"/>
2007 </xs:complexType>
2008 </xs:element>
2009 <xs:element name="USE">
2010     <xs:complexType>
2011     <xs:attribute name="handle" type="xs:string" use="required"/>
2012 </xs:complexType>
2013 </xs:element>
2014 <xs:element name="OPEN">
2015     <xs:complexType>
2016     <xs:attribute name="create" type="xs:boolean" default="true"/>
2017     <xs:attribute name="del_exist" type="xs:boolean"
2018 default="false"/>
2019     <xs:attribute name="path" type="xs:string" use="required"/>
2020 </xs:complexType>
2021 </xs:element>
2022 <xs:element name="SYSTEM">
2023     <xs:complexType>
2024     <xs:choice>

```



```

2025         <xs:element name="flush">
2026             <xs:complexType>
2027                 <xs:attribute name="handle"/>
2028                 <xs:attribute name="path"/>
2029             </xs:complexType>
2030         </xs:element>
2031     </xs:choice>
2032 </xs:complexType>
2033 </xs:element>
2034 <xs:element name="RET">
2035     <xs:complexType>
2036         <xs:choice minOccurs="0" maxOccurs="unbounded">
2037             <xs:element name="intVal" type="uoml:INT"/>
2038             <xs:element name="floatVal" type="uoml:DOUBLE"/>
2039             <xs:element name="timeVal" type="uoml:TIME"/>
2040             <xs:element name="dateVal" type="uoml:DATE"/>
2041             <xs:element name="dateTimeVal" type="uoml:DATETIME"/>
2042             <xs:element name="durationVal" type="uoml:DURATION"/>
2043             <xs:element name="stringVal" type="uoml:STRING"/>
2044             <xs:element name="binaryVal" type="uoml:BINARY"/>
2045             <xs:element name="compoundVal" type="uoml:COMPOUND"/>
2046             <xs:element name="boolVal" type="uoml:BOOL"/>
2047             <xs:element name="longVal" type="uoml:LONG"/>
2048         </xs:choice>
2049     </xs:complexType>
2050 </xs:element>
2051 <xs:complexType name="COMPOUND">
2052     <xs:annotation>
2053         <xs:documentation>compound parameter type</xs:documentation>
2054     </xs:annotation>
2055     <xs:choice minOccurs="0">
2056         <xs:element name="arc" type="uoml:ARC"/>
2057         <xs:element name="bezier" type="uoml:BEZIER"/>
2058         <xs:element name="circle" type="uoml:CIRCLE"/>
2059         <xs:element name="cmd" type="uoml:CMD"/>
2060         <xs:element name="rgb" type="uoml:COLOR_RGB"/>
2061         <xs:element name="doc" type="uoml:DOC"/>
2062         <xs:element name="docbase" type="uoml:DOCBASE"/>
2063         <xs:element name="docset" type="uoml:DOCSET"/>
2064         <xs:element name="ellipse" type="uoml:ELLIPSE"/>
2065         <xs:element name="embedfont" type="uoml:EMBEDFONT"/>
2066         <xs:element name="fontlist" type="uoml:FONTLIST"/>
2067         <xs:element name="fontmap" type="uoml:FONTMAP"/>
2068         <xs:element name="image" type="uoml:IMAGE"/>
2069         <xs:element name="layer" type="uoml:LAYER"/>
2070         <xs:element name="line" type="uoml:LINE"/>
2071         <xs:element name="matrix" type="uoml:MATRIX"/>
2072         <xs:element name="meta" type="uoml:META"/>
2073         <xs:element name="metalist" type="uoml:METALIST"/>

```

```

2074         <xs:element name="page" type="uoml:PAGE"/>
2075         <xs:element name="path" type="uoml:PATH"/>
2076         <xs:element name="rect" type="uoml:RECT"/>
2077         <xs:element name="roundrect" type="uoml:ROUNDRECT"/>
2078         <xs:element name="subpath" type="uoml:SUBPATH"/>
2079         <xs:element name="text" type="uoml:TEXT"/>
2080         <xs:element name="objstream" type="uoml:OBJSTREAM"/>
2081     </xs:choice>
2082     <xs:attribute name="name" type="xs:string"/>
2083 </xs:complexType>
2084 <xs:complexType name="STRING">
2085     <xs:annotation>
2086         <xs:documentation>string parameter type</xs:documentation>
2087     </xs:annotation>
2088     <xs:attribute name="val" type="xs:string" use="required"/>
2089     <xs:attribute name="name" type="xs:string"/>
2090 </xs:complexType>
2091 <xs:complexType name="DOUBLE">
2092     <xs:annotation>
2093         <xs:documentation>double precision float parameter
2094 type</xs:documentation>
2095     </xs:annotation>
2096     <xs:attribute name="val" type="xs:double" use="required"/>
2097     <xs:attribute name="name" type="xs:string"/>
2098 </xs:complexType>
2099 <xs:complexType name="DATE">
2100     <xs:annotation>
2101         <xs:documentation>date parameter type</xs:documentation>
2102     </xs:annotation>
2103     <xs:attribute name="val" type="xs:date" use="required"/>
2104     <xs:attribute name="name" type="xs:string"/>
2105 </xs:complexType>
2106 <xs:complexType name="DATETIME">
2107     <xs:annotation>
2108         <xs:documentation>date and time parameter
2109 type</xs:documentation>
2110     </xs:annotation>
2111     <xs:attribute name="val" type="xs:dateTime" use="required"/>
2112     <xs:attribute name="name" type="xs:string"/>
2113 </xs:complexType>
2114 <xs:complexType name="TIME">
2115     <xs:annotation>
2116         <xs:documentation>time parameter type</xs:documentation>
2117     </xs:annotation>
2118     <xs:attribute name="val" type="xs:time" use="required"/>
2119     <xs:attribute name="name" type="xs:string"/>
2120 </xs:complexType>
2121 <xs:complexType name="DURATION">
2122     <xs:annotation>

```

```

2123         <xs:documentation>duration parameter type</xs:documentation>
2124     </xs:annotation>
2125     <xs:attribute name="val" type="xs:duration" use="required"/>
2126     <xs:attribute name="name" type="xs:string"/>
2127 </xs:complexType>
2128 <xs:complexType name="BINARY">
2129     <xs:annotation>
2130         <xs:documentation>binary parameter type</xs:documentation>
2131     </xs:annotation>
2132     <xs:attribute name="val" type="xs:base64Binary" use="required"/>
2133     <xs:attribute name="name" type="xs:string"/>
2134 </xs:complexType>
2135 <xs:complexType name="INT">
2136     <xs:annotation>
2137         <xs:documentation>integer parameter type</xs:documentation>
2138     </xs:annotation>
2139     <xs:attribute name="val" type="xs:int" use="required"/>
2140     <xs:attribute name="name" type="xs:string"/>
2141 </xs:complexType>
2142 <xs:complexType name="BOOL">
2143     <xs:annotation>
2144         <xs:documentation>boolean parameter type</xs:documentation>
2145     </xs:annotation>
2146     <xs:attribute name="val" type="xs:boolean" use="required"/>
2147     <xs:attribute name="name" type="xs:string"/>
2148 </xs:complexType>
2149 <xs:complexType name="LONG">
2150     <xs:annotation>
2151         <xs:documentation>long parameter type</xs:documentation>
2152     </xs:annotation>
2153     <xs:attribute name="name" type="xs:string"/>
2154     <xs:attribute name="val" type="xs:long" use="required"/>
2155 </xs:complexType>
2156 <xs:simpleType name="CHARSTYLE">
2157     <xs:restriction base="xs:string">
2158         <xs:enumeration value="SHADOW"/>
2159         <xs:enumeration value="HOLLOW"/>
2160         <xs:enumeration value="OUTLINE"/>
2161     </xs:restriction>
2162 </xs:simpleType>
2163 </xs:schema>
2164

```

2165 **End of informative text.**

Annex B. Detailed UOML Examples

2167 **This annex is informative.**

2168 The examples below demonstrate the usage of many of the UOML instructions. Each example is followed by a
2169 corresponding "RET" instruction.

2170 The XML string of a UOML instruction may be preceded by a prolog to specify the character encoding of the
2171 XML string. If default encoding is UTF-8, the prolog, `<?xml version="1.0" encoding="UTF-8"?>`, may
2172 be omitted. The default namespace for the XML string is: `urn:oasis:names:tc:uoml:xmlns:uoml:1.0`.

2173 **Example 1: open a docbase**

2174 *Instructions sent from application to DCMS*

```
2175 <uoml:OPEN xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" create="false"
2176 del_exist="false" path="c:\test.sep"/>
```

2177 *Instructions returned from DCMS to application*

```
2178 <!-- the string value "docbase001" is the opened docbase's handle for later use -->
```

```
2179 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
```

```
2180   <boolVal name="SUCCESS" val="true"/>
```

```
2181   <stringVal name="handle" val="docbase001"/>
```

```
2182 </uoml:RET>
```

2183

2184 **Example 2 : get the root docset of the docbase (following example 1)**

2185 *Instructions sent from application to DCMS*

```
2186 <!-- since each docbase has one and only one sub-object, to get the root docset is just to
2187 get the first sub-object of docbase whose handle is returned by example 1 -->
```

```
2188 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docbase001"
2189 usage="GET_SUB">
```

```
2190   <pos val="0"/>
```

```
2191 </uoml:GET>
```

2192 *Instructions returned from DCMS to application*

```
2193 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
```

```
2194   <boolVal name="SUCCESS" val="true"/>
```

2195 <stringValue name="handle" val="docset001"/>

2196 </uoml:RET>

2197

2198 **Example 3: get the number of sub-objects of the root docset (following example 2)**

2199 *Instructions sent from application to DCMS*

2200 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docset001"
2201 usage="GET_SUB_COUNT"/>

2202 *Instructions returned from DCMS to application*

2203 <!-- the return value of 3 indicates the root docset has 3 sub-objects -->

2204 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2205 <boolVal name="SUCCESS" val="true"/>

2206 <intVal name="sub_count" val="3"/>

2207 </uoml:RET>

2208

2209 **Example 4: get the third sub-object of the docset (following example 3)**

2210 *Instructions sent from application to DCMS*

2211 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docset001"
2212 usage="GET_SUB">

2213 <pos val="2"/>

2214 </uoml:GET>

2215 *Instructions returned from DCMS to application*

2216 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2217 <boolVal name="SUCCESS" val="true"/>

2218 <stringValue name="handle" val="doc001"/>

2219 </uoml:RET>

2220 **Examples 5: get the type of a object using the empty string as the name of the property (following example 4)**

2221 *Instructions sent from application to DCMS*

2222 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP"
2223 handle="doc001">

2224 <property name=""/>

2225 </uoml:GET>
2226 *Instructions returned from DCMS to application*
2227 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2228 <boolVal name="SUCCESS" val="true"/>
2229 <stringVal name="" val="DOC"/>
2230 </uoml:RET>
2231

2232 **Example 6: get the metadata of the document (following example 4)**

2233 *Instructions sent from application to DCMS*
2234 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP"
2235 handle="doc001">
2236 <property name="metainfo"/>
2237 </uoml:GET>
2238 *Instructions returned from DCMS to application*
2239 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2240 <boolVal name="SUCCESS" val="true"/>
2241 <compoundVal name="metainfo">
2242 <metalist>
2243 <meta key="title" val="UOML Part I"/>
2244 <meta key="author" val="UOML TC"/>
2245 </metalist>
2246 </compoundVal>
2247 </uoml:RET>
2248

2249 **Example 7: get page bitmap of a page**

2250 *Instructions sent from application to DCMS*
2251 *<!-- the page object's handle is supposed to have already obtained of value "page001" in*
2252 *prior instructions(using GET) -->*
2253 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PAGE_BMP"
2254 handle="page001">

```

2255     <disp_conf addr="c:\test.bmp" end_layer="8" format="bmp" output="FILE"
2256 resolution="640">
2257     <clip>
2258         <ellipse angle="45" center="10,20" xr="30" yr="40"/>
2259         <roundrect br="70,80" tl="50,60" xr="90" yr="100"/>
2260         <subpath data="s 214,193 1 368,193 1 368,298 1 214,298"/>
2261     </clip>
2262 </disp_conf>
2263 </uoml:GET>
2264 Instructions returned from DCMS to application
2265 <!-- the bmp format of page bitmap data has been saved in the file c:\test.bmp as requested
2266 -->
2267 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2268     <boolVal name="SUCCESS" val="true"/>
2269 </uoml:RET>

```

2270

2271 **Example 8 : get first layer of a page**

```

2272 Instructions sent from application to DCMS
2273 <!-- the page object's handle is supposed to have already obtained of value "page001" in
2274 prior instructions(using GET) -->
2275 <!-- since page has only layer objects as its sub-objects, get sub-objects is the same to
2276 get layer objects -->
2277 <uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="page001"
2278 usage="GET_SUB">
2279     <pos val="0"/>
2280 </uoml:GET>
2281 Instructions returned from DCMS to application
2282 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2283     <boolVal name="SUCCESS" val="true"/>
2284     <stringVal name="handle" val="layer001"/>
2285 </uoml:RET>

```

2286

2287 **Example 9: set a text object as the current object**

2288 *Instructions send from application to DCMS*

2289 *<!-- the text object's handle is supposed to have already obtained of value "text001" in*
2290 *prior instructions(using GET) -->*

2291 `<uoml:USE xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="text001"/>`

2292 *Instructions returned from DCMS to application*

2293 `<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">`

2294 `<boolVal name="SUCCESS" val="true"/>`

2295 `</uoml:RET>`

2296

2297 **Examples 10: get spaces property of a text object (following example 9)**

2298 *Instructions send from application to DCMS*

2299 `<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP">`

2300 `<property name="spaces"/>`

2301 `</uoml:GET>`

2302 *Instructions returned from DCMS to application*

2303 `<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">`

2304 `<boolVal name="SUCCESS" val="true"/>`

2305 `<stringVal name="spaces" val="50,55"/>`

2306 `</uoml:RET>`

2307

2308 **Example 11: insert a document into a docset (following example 2)**

2309 *Instructions send from application to DCMS*

2310 `<uoml:INSERT xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docset001">`

2311 `<xobj>`

2312 `<doc name="UOML part II">`

2313 `<metainfo>`

2314 `<meta key="author" val="alex"/>`

2315 </metainfo>
2316 </doc>
2317 </xobj>
2318 </uoml:INSERT>
2319 *Instructions returned from DCMS to application*
2320 <!-- the handle of the inserted document is returned for later use -->
2321 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2322 <boolVal name="SUCCESS" val="true"/>
2323 <stringVal name="handle" val="doc002"/>
2324 </uoml:RET>
2325

2326 **Example 12: delete the document inserted in the example above**

2327 *Instructions send from application to DCMS*
2328 <uoml:DELETE xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="doc002"/>
2329 *Instructions returned from DCMS to application*
2330 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2331 <boolVal name="SUCCESS" val="true"/>
2332 </uoml:RET>
2333

2334 **Example 13: use SYSTEM to save a docbase**

2335 *Instructions send from application to DCMS*
2336 <uoml:SYSTEM xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2337 <flush path="c:\test.sep"/>
2338 </uoml:SYSTEM>
2339 <!-- instructions returned from DCMS to application -->
2340 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
2341 <boolVal name="SUCCESS" val="true"/>
2342 </uoml:RET>
2343

2344 **Example 14: close the docbase (following example 1)**

2345 *Instructions send from application to DCMS*

2346 <uoml:CLOSE xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docbase001"/>

2347 *instructions returned from DCMS to application*

2348 <uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">

2349 <boolVal name="SUCCESS" val="true"/>

2350 </uoml:RET>

2351 **End of informative text.**

2352

Annex C.RELAX NG Representation of the UOML XML Schema

2353

2354

This annex is informative.

2355

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

2356

```
default namespace = ""
```

2357

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

2358

```
start =
```

2359

```
(notAllowed
```

2360

```
| element ns1:OPEN {
```

2361

```
    attribute path { xsd:string },
```

2362

```
    attribute del_exist { xsd:boolean }?,
```

2363

```
    attribute create { xsd:boolean }?
```

2364

```
})
```

2365

```
| (notAllowed
```

2366

```
| element ns1:RET {
```

2367

```
    (element intVal { INT }
```

2368

```
    | element floatVal { DOUBLE }
```

2369

```
    | element timeVal { TIME }
```

2370

```
    | element dateVal { DATE }
```

2371

```
    | element dateTimeVal { DATETIME }
```

2372

```
    | element durationVal { DURATION }
```

2373

```
    | element stringVal { STRING }
```

2374

```
    | element binaryVal { BINARY }
```

2375

```
    | element compoundVal { COMPOUND }
```

2376

```
    | element boolVal { BOOL }
```

2377

```
    | element longVal {
```

2378

```
        attribute val { xsd:long },
```

2379

```
        attribute name { xsd:string }?
```

2380

```
    })*
```

2381

```
})
```

2382

```
| (notAllowed
```

2383

```
| element ns1:SET {
```

2384

```
    attribute handle { xsd:string }?,
```

2385

```
    (element intVal { INT }
```

2386

```
    | element floatVal { DOUBLE }
```

2387

```
    | element timeVal { TIME }
```

2388

```
    | element dateVal { DATE }
```

2389

```
    | element dateTimeVal { DATETIME }
```

2390

```
    | element durationVal { DURATION }
```

2391

```
    | element stringVal { STRING }
```

2392

```
    | element binaryVal { BINARY }
```

```

2393         | element compoundVal { COMPOUND }
2394         | element boolVal { BOOL })*
2395     })
2396 | (notAllowed
2397     | element ns1:GET {
2398         attribute handle { xsd:string }?,
2399         attribute usage { xsd:string },
2400         (element disp_conf {
2401             attribute addr { xsd:string },
2402             attribute output { xsd:string },
2403             attribute format { xsd:string }?,
2404             attribute resolution { xsd:int }?,
2405             attribute end_layer { xsd:int }?,
2406             element clip { PATH }?
2407         }
2408         | element pos {
2409             attribute val { xsd:int }
2410         }
2411         | element property {
2412             attribute name { xsd:string }
2413         })
2414     })
2415 | (notAllowed
2416     | element ns1:DELETE {
2417         attribute handle { xsd:string }?
2418     })
2419 | (notAllowed
2420     | element ns1:USE {
2421         attribute handle { xsd:string }
2422     })
2423 | (notAllowed
2424     | element ns1:INSERT {
2425         attribute pos { xsd:int }?,
2426         attribute handle { xsd:string }?,
2427         element xobj { COMPOUND }
2428     })
2429 | (notAllowed
2430     | element ns1:SYSTEM {
2431         element flush {
2432             attribute path { text }?,
2433             attribute handle { text }?
2434         }
2435     })
2436 | (notAllowed
2437     | element ns1:CLOSE {
2438         attribute handle { xsd:string }?
2439     })
2440 COMPOUND =
2441     (attribute name { xsd:string }?,

```

```

2442 ((notAllowed
2443 | element arc {
2444     attribute angle { xsd:float },
2445     attribute center { xsd:string },
2446     attribute end { xsd:string },
2447     attribute start { xsd:string },
2448     attribute clockwise { xsd:boolean }
2449 })
2450 | (notAllowed
2451 | element bezier {
2452     attribute end { xsd:string },
2453     attribute ctrl2 { xsd:string }?,
2454     attribute ctrl { xsd:string },
2455     attribute start { xsd:string }
2456 })
2457 | (notAllowed
2458 | element circle { CIRCLE })
2459 | (notAllowed
2460 | element cmd {
2461     attribute v2 {
2462         text
2463         # <data type="anySimpleType"/>
2464
2465     }?,
2466     attribute v1 {
2467         text
2468         # <data type="anySimpleType"/>
2469
2470     }?,
2471     attribute name {
2472         xsd:string "CHAR_WEIGHT"
2473         | xsd:string "CLIP_AREA"
2474         | xsd:string "COLOR_FILL"
2475         | xsd:string "CHAR_SIZE"
2476         | xsd:string "LINE_CAP"
2477         | xsd:string "SHADOW_LEN"
2478         | xsd:string "CHAR_STYLE"
2479         | xsd:string "RENDER_MODE"
2480         | xsd:string "CHAR_SLANT"
2481         | xsd:string "COLOR_LINE"
2482         | xsd:string "TEXT_DIR"
2483         | xsd:string "COLOR_TEXT"
2484         | xsd:string "GRAPH_MATRIX"
2485         | xsd:string "HOLLOW_BORDER"
2486         | xsd:string "POP_GS"
2487         | xsd:string "PUSH_GS"
2488         | xsd:string "LINE_WIDTH"
2489         | xsd:string "CHAR_DIR"
2490         | xsd:string "OUTLINE_WIDTH"

```

```

2491         | xsd:string "FILL_RULE"
2492         | xsd:string "EXT_MATRIX"
2493         | xsd:string "SHADOW_WIDTH"
2494         | xsd:string "RASTER_OP"
2495         | xsd:string "TEXT_MATRIX"
2496         | xsd:string "LINE_JOIN"
2497         | xsd:string "SHADOW_NEG"
2498         | xsd:string "SHADOW_ATL"
2499         | xsd:string "CHAR_ROTATE"
2500         | xsd:string "MITER_LIMIT"
2501         | xsd:string "COLOR_OUTLINE"
2502         | xsd:string "FONT"
2503         | xsd:string "IMAGE_MATRIX"
2504         | xsd:string "SHADOW_DIR"
2505         | xsd:string "OUTLINE_BORDER"
2506         | xsd:string "COLOR_SHADOW"
2507     },
2508     (element cliparea { PATH }
2509     | element matrix { MATRIX }
2510     | element rgb { COLOR_RGB })?
2511 )
2512 | (notAllowed
2513     | element rgb { COLOR_RGB })
2514 | (notAllowed
2515     | element doc {
2516         attribute name { xsd:string },
2517         element metainfo { METALIST }
2518     })
2519 | (notAllowed
2520     | element docbase {
2521         attribute path { xsd:string },
2522         attribute name { xsd:string }
2523     })
2524 | (notAllowed
2525     | element docset {
2526         attribute name { xsd:string }
2527     })
2528 | (notAllowed
2529     | element ellipse { ELLIPSE })
2530 | (notAllowed
2531     | element embedfont { xsd:base64Binary })
2532 | (notAllowed
2533     | element fontlist { empty })
2534 | (notAllowed
2535     | element fontmap {
2536         attribute no { xsd:int },
2537         attribute name { xsd:string }
2538     })
2539 | (notAllowed

```

```

2540     | element image {
2541         attribute tl { xsd:string },
2542         attribute br { xsd:string },
2543         attribute type { xsd:string },
2544         attribute path { xsd:string }?,
2545         xsd:base64Binary
2546     })
2547 | (notAllowed
2548     | element layer { empty })
2549 | (notAllowed
2550     | element line {
2551         attribute end { xsd:string },
2552         attribute start { xsd:string }
2553     })
2554 | (notAllowed
2555     | element matrix { MATRIX })
2556 | (notAllowed
2557     | element meta { META })
2558 | (notAllowed
2559     | element metalist { METALIST })
2560 | (notAllowed
2561     | element page {
2562         attribute resolution { xsd:int },
2563         attribute height { xsd:float },
2564         attribute width { xsd:float }
2565     })
2566 | (notAllowed
2567     | element path { PATH })
2568 | (notAllowed
2569     | element rect { RECT })
2570 | (notAllowed
2571     | element roundrect { ROUNDRECT })
2572 | (notAllowed
2573     | element subpath { SUBPATH })
2574 | (notAllowed
2575     | element text {
2576         attribute spaces { xsd:string }?,
2577         attribute text { xsd:string },
2578         attribute encode { xsd:string },
2579         attribute origin { xsd:string }
2580     })
2581 | (notAllowed
2582     | element objstream { empty })))?),
2583 empty
2584 PATH =
2585 ((notAllowed
2586     | element subpath { SUBPATH })
2587 | (notAllowed
2588     | element rect { RECT })

```

```

2589     | (notAllowed
2590         | element circle { CIRCLE })
2591     | (notAllowed
2592         | element ellipse { ELLIPSE })
2593     | (notAllowed
2594         | element roundrect { ROUNDRECT }))**,
2595     empty
2596 METALIST =
2597     (notAllowed
2598         | element meta { META })**,
2599     empty
2600 COLOR_RGB =
2601     (attribute a { xsd:short }?,
2602     attribute b { xsd:short },
2603     attribute g { xsd:short },
2604     attribute r { xsd:short }),
2605     empty
2606 TIME =
2607     (attribute name { xsd:string }?,
2608     attribute val { xsd:time }),
2609     empty
2610 ELLIPSE =
2611     (attribute angle { xsd:float },
2612     attribute center { xsd:string },
2613     attribute yr { xsd:int },
2614     attribute xr { xsd:int }),
2615     empty
2616 SUBPATH =
2617     attribute data { xsd:string },
2618     empty
2619 INT =
2620     (attribute name { xsd:string }?,
2621     attribute val { xsd:int }),
2622     empty
2623 DURATION =
2624     (attribute name { xsd:string }?,
2625     attribute val { xsd:duration }),
2626     empty
2627 ROUNDRECT =
2628     (attribute br { xsd:string },
2629     attribute tl { xsd:string },
2630     attribute yr { xsd:int },
2631     attribute xr { xsd:int }),
2632     empty
2633 DATE =
2634     (attribute name { xsd:string }?,
2635     attribute val { xsd:date }),
2636     empty
2637 BINARY =

```



```
2638     (attribute name { xsd:string }?,
2639     attribute val { xsd:base64Binary }),
2640     empty
2641 STRING =
2642     (attribute name { xsd:string }?,
2643     attribute val { xsd:string }),
2644     empty
2645 DOUBLE =
2646     (attribute name { xsd:string }?,
2647     attribute val { xsd:double }),
2648     empty
2649 BOOL =
2650     (attribute name { xsd:string }?,
2651     attribute val { xsd:boolean }),
2652     empty
2653 CIRCLE =
2654     (attribute center { xsd:string },
2655     attribute radius { xsd:int }),
2656     empty
2657 META =
2658     (attribute val { xsd:string },
2659     attribute key { xsd:string }),
2660     empty
2661 MATRIX =
2662     (attribute f32 { xsd:float },
2663     attribute f31 { xsd:float },
2664     attribute f22 { xsd:float },
2665     attribute f21 { xsd:float },
2666     attribute f12 { xsd:float },
2667     attribute f11 { xsd:float }),
2668     empty
2669 RECT =
2670     (attribute br { xsd:string },
2671     attribute tl { xsd:string }),
2672     empty
2673 DATETIME =
2674     (attribute name { xsd:string }?,
2675     attribute val { xsd:dateTime }),
2676     empty
```

2677

2678

2679 **End of informative text.**

2680

2682

Annex D.Acknowledgements

2683

2684

2685 **This annex is informative.**

2686 The following individuals have participated in the creation of this specification and are gratefully acknowledged:

2687

2688 **Participants:**

2689 Alex Wang, Sursen Corporation

2690 Xu Guo, Sursen Corporation

2691 Ningsheng Liu, Sursen Corporation

2692 Allison Shi, Sursen Corporation

2693 Stephen Green, Individual

2694 Kaihong Zou, Sursen Corporation

2695 Pine Zhang, UOML Alliance

2696 Mendy Liu, UOML Alliance

2697 Joel Marcey, Sursen Corporation

2698 Andy Li, Changfeng Open Standards Platform Software Alliance

2699 Charles H. Schulz, Ars Aperta

2700 Lin Cheng, Beijing Redflag CH2000 Software Co. Ltd.

2701 Liwei Wang, Sursen Corporation

2702

2703 **End of informative text.**