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.

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

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

Status:

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

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

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

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

Notices

Copyright © OASIS® 2010. All Rights Reserved.

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

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

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

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

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

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

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

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

Table of Contents

1. Introduction **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

Copyright© OASIS® 1993-2010. All Rights Reserved.

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

This text is informative

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

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

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

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

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

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

End of informative text

1.1 Terminology

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

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

DCMS: Abbreviated for “Docbase Management System”.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
<CMD name="COLOR_LINE" >
<rgb r="128" g="3" b="255" a="120"/>
</CMD>
```

where rgb is a sub element of CMD.

end note]

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

```
docbase
  docset
    document
       page
         layer
            object stream
```

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

UOML: abbreviation of "Unstructured Operation Markup Language".

1.2 Scope

This OASIS standard describes the abstract document model of UOML and the operations available on it. Specifically, operations providing functionality for read/write/edit and display/print on layout-based documents are described.

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

1.3 Notational Conventions

The following typographical conventions are used in this OASIS standard:

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

1.4 Acronyms and Abbreviations

This clause is informative

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

DCMS — Docbase Management System

IEC — the International Electrotechnical Commission

ISO — the International Organization for Standardization

UOML — Unstructured Operation Markup Language

W3C — World Wide Web Consortium

End of informative text

1.5 General Description

This OASIS standard is divided into the following subdivisions:

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

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

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

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

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

- Introductory text in clause 1
- Sub-clauses 1.4, 1.6, and 1.8
- All annexes
- All notes and examples

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

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

1.6 Overview

This clause is informative

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

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

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

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

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

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

End of informative text

1.7 Normative References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.8 Non-Normative References

This clause is informative.

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

End of informative text.

2. Abstract Document Model

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

2.1 Overview

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

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

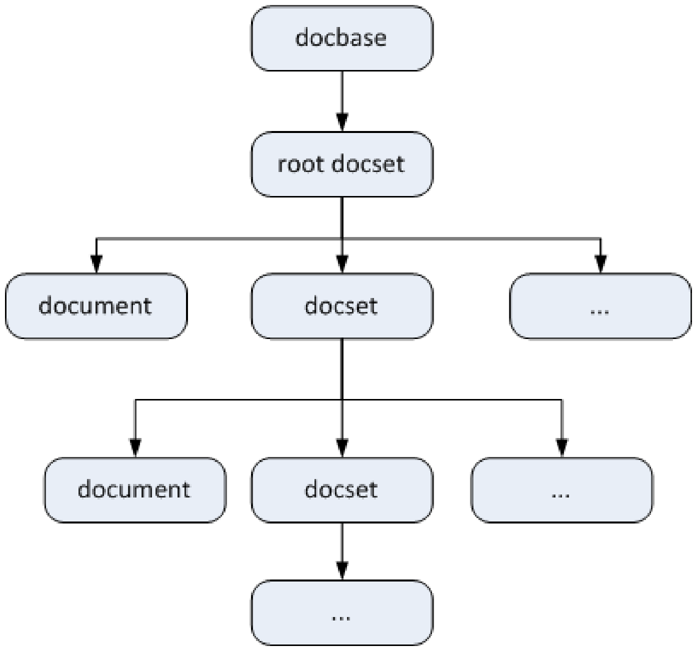


Figure1. UOML Abstract document Model 1

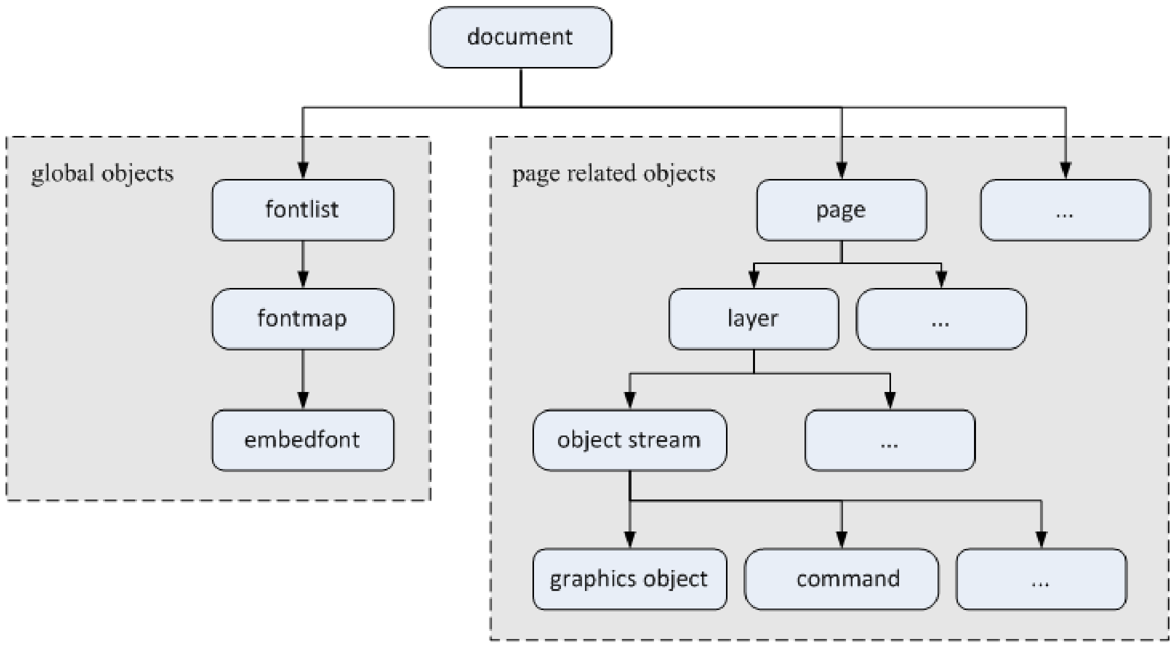


Figure 2. UOML Abstract Document Model 2

The following clauses provide a description of each object type.

2.2 Docbase

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

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

Sub-object: docset.

2.3 Docset

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

Sub-object: document, docset.

2.4 Document

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

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

Sub-object: fontlist, page.

2.5 Font

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

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

2.6 Page

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

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

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

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

Sub-object: layer.

2.7 Layer

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

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

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

Sub-object: object stream.

2.8 Object Stream

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

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

Sub-object: graphics object, command object.

2.9 Graphics Object

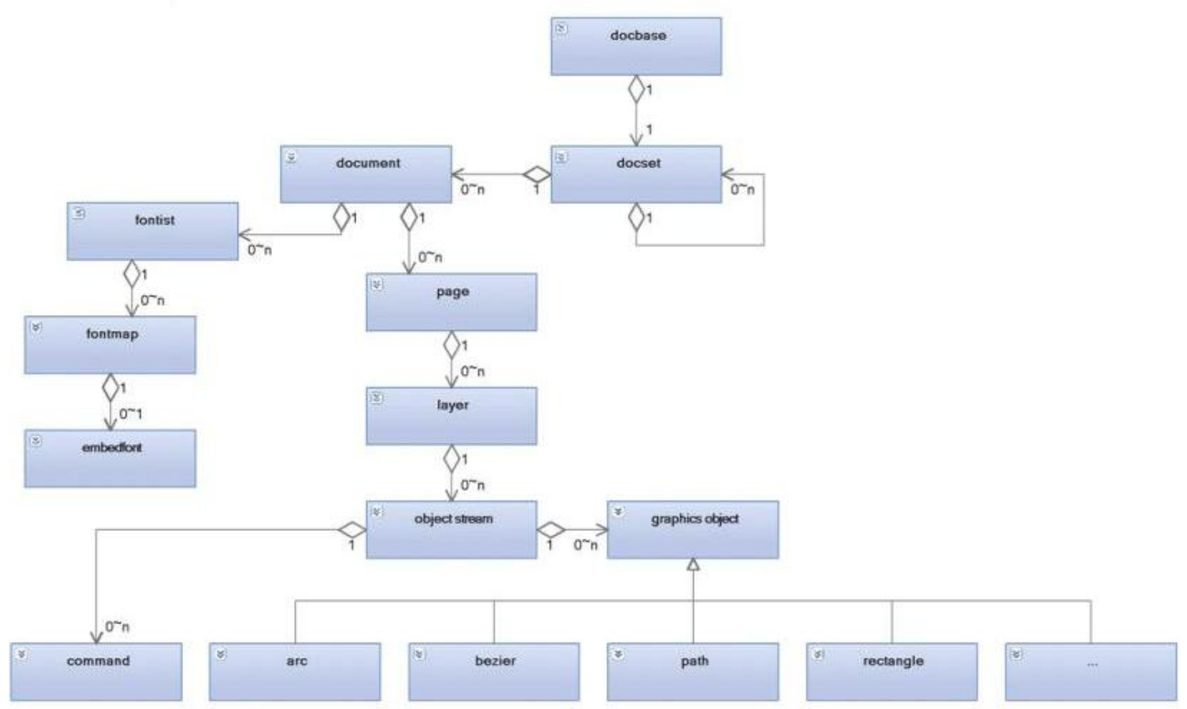
A graphics object is a set of objects that could allow the render engine to draw text, image, and Path. Graphics objects describe the appearance of the page. The graphics objects in UOML includes arc, Bezier, circle, ellipse, image, line, rectangle, round rectangle, Path and text objects.

2.10 Command Object

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

2.11 UML Diagram of UOML

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



Copyright© OASIS® 1993-2010. All Rights Reserved.

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.

3. UOML Instructions

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

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

3.1 OPEN

Semantics:

OPEN creates or opens a docbase.

Properties:

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

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

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

Sub-elements: N/A

Return value:

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

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

[*Example*:

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

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

Return element if OPEN succeeds:

<uoml:RET>
      <boolVal name="SUCCCESS" val="true"/>
      <stringVal name="HANDLE" val="db_handle_xxxxx"/>
```

```
</uoml:RET>

Return element if OPEN fails:

<uoml:RET>
      <boolVal name="SUCCCESS" val="false"/>
      <stringVal name="ERR_INFO" val="required resource not available"/>
</uoml:RET>
```

end example]

3.2 CLOSE

Semantics:

CLOSE closes a docbase

Properties:

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

Sub-elements: N/A

Return value:

Defined by RET

[*Example*:

Close a docbase.

```
<uoml:CLOSE handle="db_handle_xxxxx"/>
```

end example]

3.3 USE

Semantics:

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

Properties:

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

Sub-elements: N/A

Return value:

Defined by RET

[*Example*:

Set up the handle represented object as the current object.

```
<uoml:USE handle="obj_handle_xxxxxx"/>
```

end example]

3.4 GET

Semantics:

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

Properties:

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

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

Sub-elements:

pos: used when usage=GET_SUB.

Property of this sub-element:

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

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

property: used when usage=GET_PROP.

Property of this sub-element:

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

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

disp_conf: used when usage=GET_PAGE_BMP.

Properties of this sub-element:

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

resolution: represents resolution of bitmap

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

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

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

Sub-element of this sub-element:

clip: represents clip area for output, PATH type.

Usage value / Return value:

The return value is based on the usage value:

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

[*Example*:

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

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

RET instruction returns the number:

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

end example]

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

[*Example*:

Get a specific sub-object handle:

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

RET instruction returns the handle of the sub-object:

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

end example]

- GET_PROP: If the usage is GET_PROP, this indicates to get some specific property of a specific object. If the name property is a non-empty string, GET shall contain the sub-element of 'property'. If the operation succeeds, the sub-element of return value, which is returned via RET instruction, is variant; the sub-element name relies on the type it has retrieved, the 'name' property of the sub-element is the property name to get, 'val' property is the value of the property; otherwise if the name property is an empty string, the RET instruction returns a stringVal value representing the type of the object, which is the element name of the XML description of the object without the namespace prefix.

[*Example*:

Get specific property of the object

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

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

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

end example]

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

[*Example*:

Get specific page's bitmap

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

end example]

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

[*Example*:

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

end example]

3.5 SET

Semantics:

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

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

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

Properties:

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

Sub-element:

intVal: set up integer type value, INT type

floatVal: set up float type value, DOUBLE type.

timeVal: set up time value, TIME type.

dateVal: set up date value, DATE type.

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

durationVal: set up time duration value, DURATION type.

stringVal: set up string type value, STRING type.

binaryVal: set up binary type value, BINARY type.

compoundVal: set up compound type value, COMPOUND type.

boolVal: set up boolean type value, BOOLEAN type.

Return value:

defined by RET.

[*Example*:

Set specific object's angle property.

```
<uoml:SET handle="obj_handle_xxxxxx">
    <floatVal name="angle" val="0.1"/>
</uoml:SET>
```

end example]

3.6 INSERT

Semantics:

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

Properties:

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

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

Sub-element:

xobj: xml expression of the sub-object.

Return value:

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

[*Example*:

Insert text data

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

end example]

[*Example*:

Insert a layer

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

end example]

3.7 DELETE

Semantics:

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

Properties:

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

Sub-element: N/A

Return value:

Defined by RET

[*Example*:

Delete an object

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

end example]

3.8 SYSTEM

Semantics:

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

Properties:

N/A

Sub-element:

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

Return value:

Defined by RET

[*Example*:

Save the docbase example.sep

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

end example]

3.9 RET

Semantics:

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

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

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

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

RET contains at least one 'boolVal' sub-element to describe whether the operation was successful or not. Its 'name' property is SUCCESS, and its 'val' property is either ‘true’ or ‘false’, depending on the success of the operation.

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

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

Properties: N/A

Sub-element:

intVal: integer type return value, INT type

floatVal: float type return value, DOUBLE type.

TimeVal: time type return value, TIME type.

DateVal: date type return value, DATE type.

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

DurationVal: time duration type return value, DURATION type.

StringVal: string type return value, STRING type.

BinaryVal: binary type return value, BINARY type.

CompoundVal: compound type return value, COMPOUND type.

BoolVal: boolean type return value, BOOLEAN type.

[*Example*:

Return two values.

```
  <uoml:RET>
    <boolVal name="SUCCESS" val="false"/>
    <stringVal name="ERR_INFO" val="required resource not available"/>
  </uoml:RET>
```

end example]

4. UOML Objects

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

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

4.1 Logical Coordinate System and Units

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

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

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

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

4.2 Color Model

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

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

4.3 Graphics State

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

4.4 DOCBASE

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

Properties:

name: name of docbase.

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

Sub-elements: N/A

4.5 DOCSET

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

Properties:

name: name of docset.

Sub-elements: N/A

4.6 DOC

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

Properties:

name: name of document.

Sub-elements:

metainfo: metadata of the document, METALIST type.

4.6.1 Metadata

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

4.6.1.1 METALIST

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

Properties: N/A

Sub-elements:

meta: META type.

4.6.1.2 META

Semantics: One item of metadata.

Properties:

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

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

Sub-elements: N/A

4.7 FONT DEFINITION

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

4.7.1 FONTLIST

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

Properties: N/A

Sub-elements: N/A

4.7.2 FONTMAP

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

Properties:

name: name of the font

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

Sub-elements: N/A

4.7.3 EMBEDFONT

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

Properties: N/A

Sub-elements: N/A

4.8 PAGE

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

Properties:

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

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

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

Sub-elements: N/A

4.9 LAYER

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

Properties: N/A

Sub-elements: N/A

4.10 OBJSTREAM

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

Properties: N/A

Sub-elements: N/A

4.11 Graphics Objects

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

4.11.1 ARC

Semantics:

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

Properties:

start: starting position of the arc.

end: ending position of the arc.

center: center of the arc's ellipse.

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

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

Sub-elements: N/A

4.11.2 BEZIER

Semantics:

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

Properties:

start: starting point of the Bezier curve.

ctrl: the first control point of the Bezier curve.

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

end: ending point of the Bezier curve.

Sub-elements: N/A

4.11.3 CIRCLE

Semantics:

A circle, specified by a center and radius.

Properties:

center : coordinate of the circle center.

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

Sub-elements: N/A

4.11.4 ELLIPSE

Semantics:

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

Properties:

center: coordinates of ellipse center.

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

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

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

Sub-elements: N/A

4.11.5 IMAGE

Semantics:

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

Copyright© OASIS® 1993-2010. All Rights Reserved.

Properties:

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

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

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

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

Sub-elements: N/A

Sub-objects: N/A

4.11.6 LINE

Semantics:

A line, specified by a starting and ending point.

Properties:

start: coordinates of where the line starts.

end: coordinates of where the line ends.

Sub-elements: N/A

4.11.7 RECT

Semantics:

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

Properties:

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

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

Sub-elements: N/A

4.11.8 ROUNDRECT

Semantics:

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

Properties:

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

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

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

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

Sub-elements: N/A

4.11.9 SUBPATH

Semantics:

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

Properties:

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

Sub-elements: N/A

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

```
<uoml:INSERT pos=”2” handle=”vs03”>
 <xobj>
  <path>
   <subpath data="s 214,193 l 368,193 l 368,298 l 214,298"/>
   <subpath data="s 417,206 B 417,186 426,167 435,167 B 443,167 452,230 452,293"/>
  </path>
 </xobj>
</uoml:INSERT>
```

end example].

4.11.10 PATH

Semantics:

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

Properties: N/A

Sub-elements:

circle: CIRCLE type, defines a circle.

ellipse: ELLIPSE type, defines an ellipse.

rect: RECT type, defines a rectangle.

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

subpath: SUBPATH type, defines a subpath.

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

```
        <uoml:INSERT pos="4">
          <xobj>
           <path>
            <circle center="167,251" radius="70" />
             <rect tl="124,135" br="345,257"/>
           </path>
          </xobj>
        </uoml:INSERT>

```

end example].

4.11.11 TEXT

Semantics:

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

Properties:

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

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

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

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

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

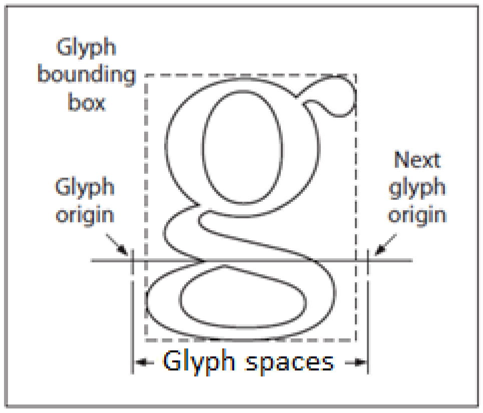


Figure 4. spaces of text

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

Sub-elements: N/A

4.11.12 Coordinate and subpath Encoding Rules

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

Coordinate encoding rules

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

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

Path encoding rules

```
path = start , { blank , ( line | bezier2 | bezier3 | arc ) } ;
start = 's' , blank , coord ;
line = 'l' , blank , coord ;
bezier2 = 'b' , blank , coord , blank , coord ;
bezier3 = 'B' , blank , coord , blank , coord , blank , coord ;
arc = 'a' , blank , clockwise , blank , angle , blank , coord , blank , coord ;
clockwise = 'true' | 'false' ;
angle = float ;
number = [ '-' ] , digit , { digit } ;
float = number [ , '.' , { digit } ][ , ( 'e' | 'E' ) , [(‘+’|’-‘)] , digit , {digit}] ;
digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
blank = ' ' , { ' ' } ;
```

Semantics

"`coord`" represents coordinates.

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

"`line`" represents a line segment.

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

"`bezier3`" represents third-order Bezier curve.

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

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

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

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

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

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

4.12 Command Object

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

4.12.1 CMD

Semantics: XML description of command objects.

Properties:

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

v1: optional command value.

v2: optional command value.

Sub-elements:

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

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

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

Sub-objects: N/A

[*Example*:

```
  <uoml:INSERT pos=”2” handle=”vs03”>
    <xobj>
      <cmd name="COLOR_LINE" >
        <rgb r="128" g="3" b="255" a="120"/>
      </cmd>
    </xobj>
</uoml:INSERT>
```

end example]

[*Example*:

```
  <uoml:INSERT pos=”2” handle=”vs03”>
    <xobj>
      <cmd name="LINE_CAP" v1="END_BUT"/>
    </xobj>
  </uoml:INSERT>
```

end example]

[*Example*:

```
  <uoml:INSERT pos=”2” handle=”vs03”>
    <xobj>
      <cmd name="TEXT_MATRIX">
        <matrix f11="2" f12="0" f21="0" f22="1.5" f31="10" f32="20"/>
      </cmd>
    </xobj>
  </uoml:INSERT>
```

end example]

4.12.2 Values for CMD’s ‘name’ property

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

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

4.12.2.1 COLOR_LINE

Semantics: Set the current line color

Properties: N/A

Sub-elements:

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

4.12.2.2 COLOR_FILL

Semantics: Set the current fill color

Properties: N/A

Sub-elements:

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

4.12.2.3 COLOR_SHADOW

Semantics: Set the current character shadow color

Properties: N/A

Sub-elements:

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

4.12.2.4 COLOR_OUTLINE

Semantics: Set the current character outline color

Properties: N/A

Sub-elements:

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

4.12.2.5 COLOR_TEXT

Semantics: Set the current text color

Properties: N/A

Sub-elements:

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

4.12.2.6 LINE_WIDTH

Semantics: set the current line width/thickness

Properties:

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

Sub-elements: N/A

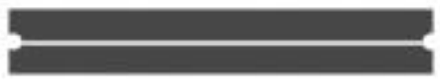
4.12.2.7 LINE_CAP

Semantics: Set the current line cap style

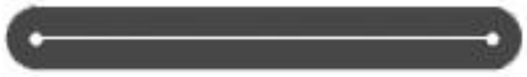
Properties:

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

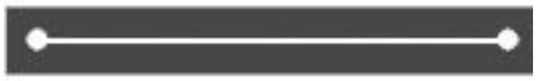
END_BUT: the stroke shall be squared off at the endpoint of the path. There shall be no projection beyond the end of the path.



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.



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.



Sub-elements: N/A

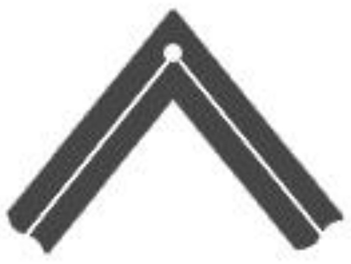
4.12.2.8 LINE_JOIN

Semantics: Set the current line join style

Properties:

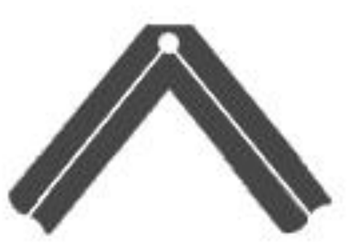
v1: a character string, representing the line join style. Possible values for this property are JOIN_MITER, JOIN_BEVEL and JOIN_ROUND

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

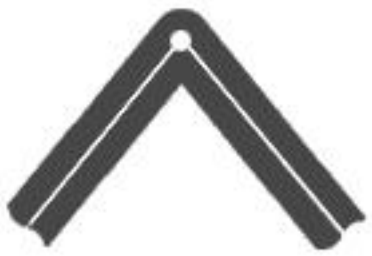


JOIN_BEVEL: the two segments shall be finished with END_BUT and the resulting notch beyond the end of the segments shall be filled with a triangle.

Copyright© OASIS® 1993-2010. All Rights Reserved.



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.



Sub-elements: N/A

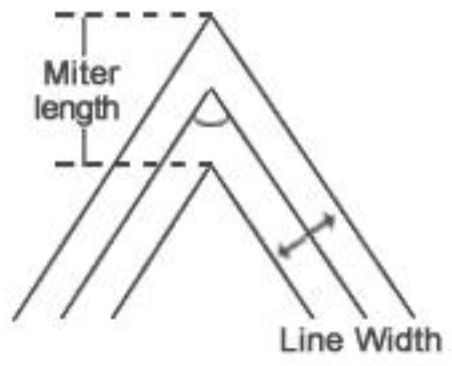
4.12.2.9 MITER_LIMIT

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.

Properties:

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

Sub-elements: N/A



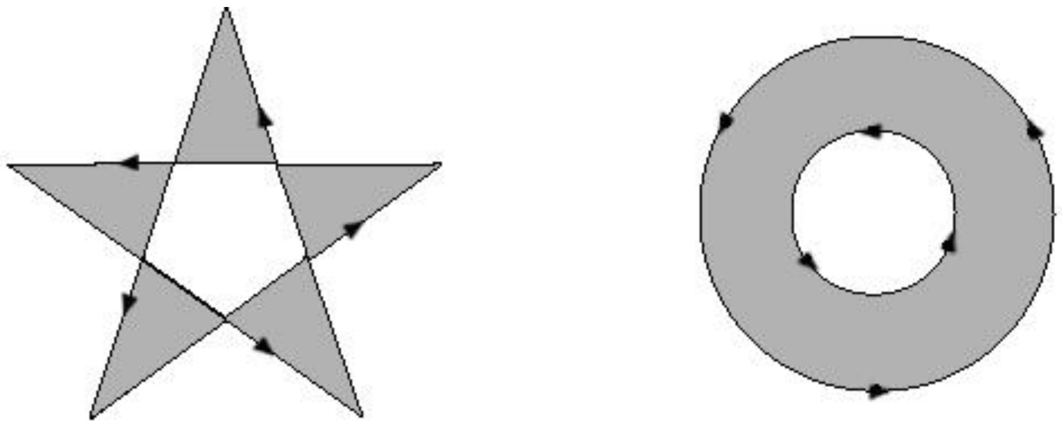
4.12.2.10 FILL_RULE

Semantics: Set the current fill rules

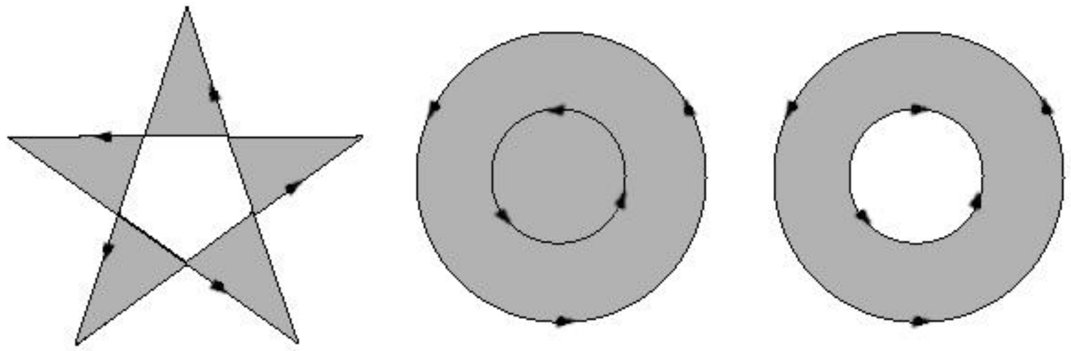
Properties:

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

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



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



Sub-elements: N/A

Note:

4.12.2.11 RENDER_MODE

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

Properties:

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

LINE: draw a line along the path.

FILL: draw the entire region enclosed by the path.

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

Sub-elements: N/A

4.12.2.12 RASTER_OP

Semantics: Set the current raster operation.

Properties:

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

ROP_COPY: pixel_color = src

ROP_N_COPY: pixel_color = ~src

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

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

ROP_NOP: pixel_color = dest

ROP_REV: pixel_color = ~dest

ROP_AND: pixel_color = src & dest

ROP_AND_N: pixel_color = src & ~dest

ROP_N_AND: pixel_color = ~src & dest

ROP_N_AND_N: pixel_color = ~src & ~dest

ROP_OR: pixel_color = src | dest

ROP_OR_N: pixel_color = src | ~dest

ROP_N_OR: pixel_color = ~src | dest

ROP_N_OR_N: pixel_color = ~src | ~dest

ROP_XOR: pixel_color = src ^ dest

ROP_EOR: pixel_color = src ^ ~dest

Sub-elements: N/A

4.12.2.13 TEXT_DIR

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

Properties:

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

Sub-elements: N/A

4.12.2.14 CHAR_DIR

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

Properties:

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

Sub-elements: N/A

4.12.2.15 CHAR_ROTATE

Semantics: Set the current character rotation angle.

Properties:

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

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

Sub-elements: N/A

4.12.2.16 CHAR_SLANT

Semantics: Set the slant of the character.

Properties:

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

Sub-elements: N/A

4.12.2.17 CHAR_SIZE

Semantics: Set the current character width and height.

Properties:

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

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

Sub-elements: N/A

4.12.2.18 CHAR_WEIGHT

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

Properties:

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

Sub-elements: N/A

4.12.2.19 CHAR_STYLE

Semantics: Set the current character style.

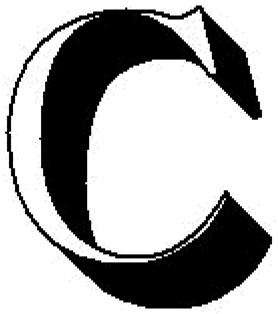
Properties:

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

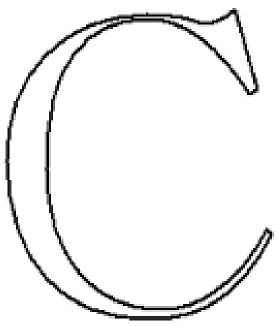
SHADOW: set shadow style. If this character style is set, then the following algorithm is used to render the shadow effect:

- If SHADOW_NEG (§4.11.2.30) is false, the character is extended with a distance of SHADOW_LEN (§4.11.2.27) along the shadow direction (§4.11.2.28), then a hollowed character with raster operation ROP_COPY is drawn in the original position. The border width of the hollowed character is SHADOW_WIDTH (§4.11.2.26).

- If SHADOW_NEG is true, the character position is moved with a distance of SHADOW_LEN along the shadow direction, and extended SHADOW_WIDTH along the shadow direction; then the character is drawn in the original position with background color and raster operation ROP_COPY, and extended with a distance SHADOW_LEN along the shadow direction; then in the original position, a character with normal color and raster operation ROP_COPY is drawn.



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

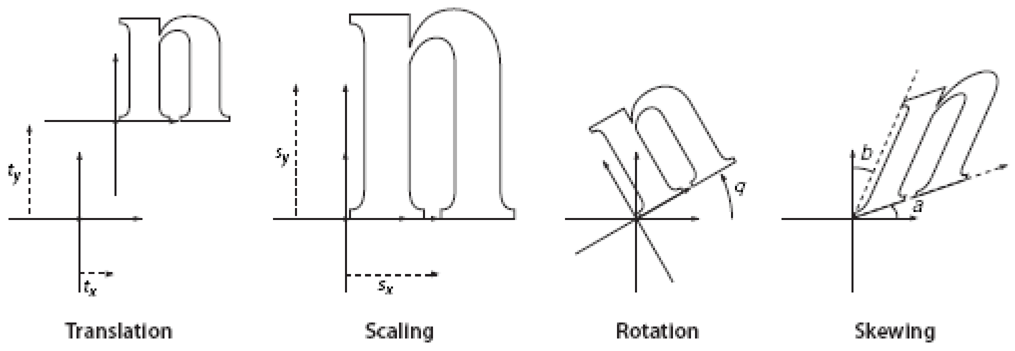


OUTLINE: set outline style. If this character style is set, a line with thickness OUTLINE_BORDER (§4.11.2.33), and with distance OUTLINE_WIDTH (§4.11.2.34) from the outline of the character, should be drawn along the outline of the character.

Sub-elements: N/A

4.12.2.20 TEXT_MATRIX

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



Properties: N/A

Sub-elements:

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

4.12.2.21 IMAGE_MATRIX

Semantics: Set the current image transformation matrix

Properties: N/A

Sub-elements:

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

4.12.2.22 GRAPH_MATRIX

Semantics: Set the current line/curve transformation matrix

Properties: N/A

Sub-elements:

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

4.12.2.23 EXT_MATRIX

Semantics: Set the current extension transformation matrix

Properties: N/A

Sub-elements:

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

4.12.2.24 PUSH_GS

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

Properties: N/A

Sub-elements: N/A

4.12.2.25 POP_GS

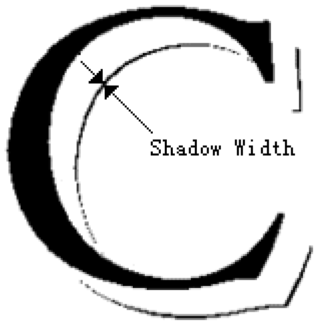
Semantics: Pop out the top value from the graphics state stack, replacing the current graphics state.

Properties: N/A

Sub-elements: N/A

4.12.2.26 SHADOW_WIDTH

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



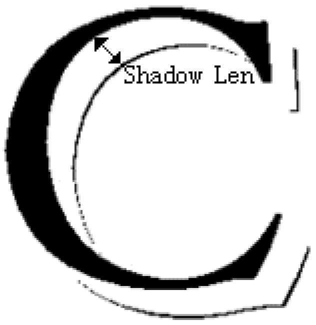
Properties:

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

Sub-elements: N/A

4.12.2.27 SHADOW_LEN

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



Properties:

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

Sub-elements: N/A

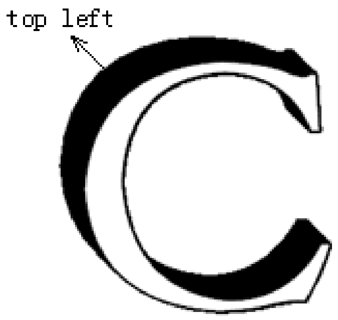
4.12.2.28 SHADOW_DIR

Semantics: Set the direction of the current character shadow

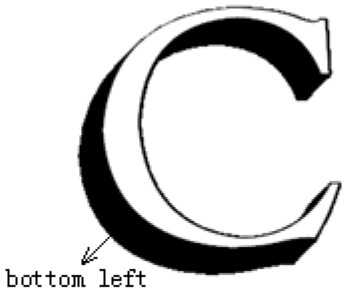
Properties:

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

SHADOW_LT: the character shadow direction is top left.

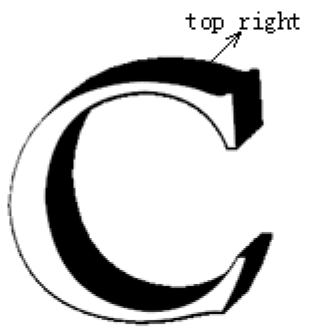


SHADOW_LB: the character shadow direction is bottom left.

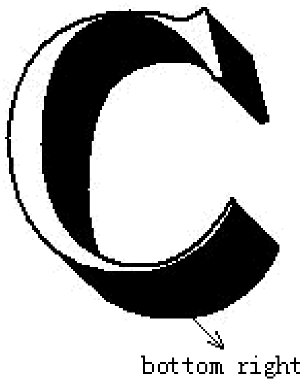


SHADOW_RT: the character shadow direction is top right.

Copyright© OASIS® 1993-2010. All Rights Reserved.



SHADOW_RB: the character shadow direction is bottom right.



Sub-elements: N/A

4.12.2.29 SHADOW_ATL

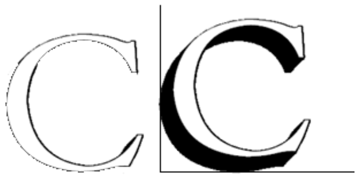
Semantics: Set whether to adjust the coordinates of a character when the direction of character shadow is to the left or bottom.

Properties:

v1: a Boolean value, representing whether to alter the coordinates of a character. The value ‘true’ specifies that the coordinates are altered.

Sub-elements: N/A

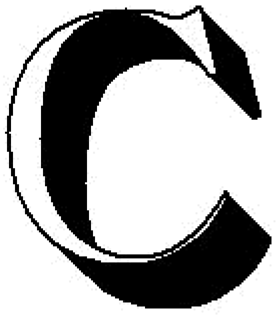
[*Example*: Illustrated in the figures below, when a character is shadowed, the bounding box of its outline is bigger. If two characters that are not shadowed are adjacent, their baselines are aligned horizontally. A shadow effect will break this horizontal alignment. Also, a shadow to the left will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



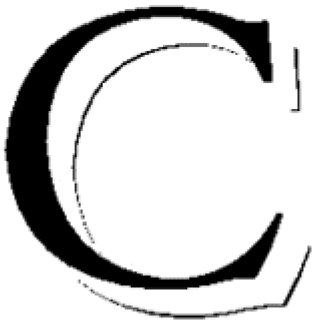
end example]

4.12.2.30 SHADOW_NEG

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



SHADOW_NEG is false



SHADOW_NEG is true

Properties:

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

Sub-elements: N/A

4.12.2.31 CLIP_AREA

Semantics: Set the current clip area

Properties: N/A

Sub-elements:

cliparea: PATH type, representing the new clip area.

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

4.12.2.32 FONT

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

Properties:

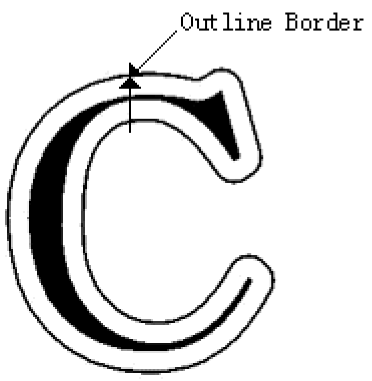
v1: a character string, representing the encoding/character set. The valid value for this property is the same as for the *encode* property of TEXT (§4.10.11).

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

Sub-elements: N/A

4.12.2.33 OUTLINE_BORDER

Semantics: Set the border width of the current outline character



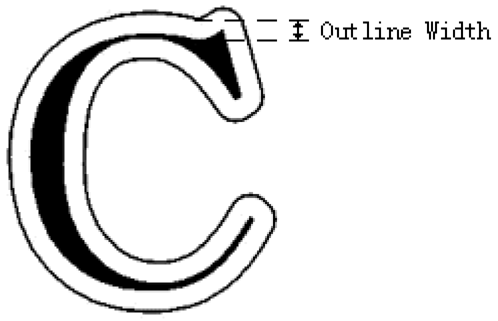
Properties:

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

Sub-elements: N/A

4.12.2.34 OUTLINE_WIDTH

Semantics: Set the outline width of the current outline character



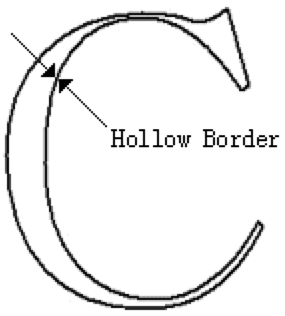
Properties:

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

Sub-elements: N/A

4.12.2.35 HOLLOW_BORDER

Semantics: Set the border width of the current hollow character



Properties:

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

Sub-elements: N/A

4.12.3 Definition of Referenced Type

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

4.12.3.1 COLOR_RGB

Semantics: the value of a color setting

Properties:

r: red component

g: green component

b: blue component

a: optional alpha component.

Sub-element: N/A

4.12.3.2 MATRIX

Semantics: the values in a transformation matrix

Properties:

f11: floating point number

f12: floating point number

f21: floating point number

f22: floating point number

f31: floating point number

f32: floating point number

Sub-element: N/A

[*Note*:

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

x′ = f11×x + f21×y + f31
y′ = f12×x + f22×y + f32

• Translations are specified using [1 0 0 1 *tx ty*], where *tx* and *ty* shall be the distances to translate the origin of the coordinate system in the horizontal and vertical dimensions, respectively.
• Scaling is specified using [*sx* 0 0 *sy* 0 0]. This scales the coordinates so that 1 unit in the horizontal and vertical dimensions of the new coordinate system is the same size as *sx* and *sy* units, respectively, in the previous coordinate system.
• Rotations are specified using by [cos(*q*) sin(*q*) –sin(*q*) cos(*q*) 0 0], which has the effect of rotating the coordinate system axes by an angle *q* counterclockwise.
• 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 an angle *b*.

end note]

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

4.14 Definition of Parameter Data Types

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

4.14.1 INT

Properties:

name: a character string value, xs:string type

val: xs:integer type

Sub-element: N/A

4.14.2 DOUBLE

Properties:

name: a character string, xs:string type

val: xs:double type

Sub-element: N/A

4.14.3 LONG

Properties:

name: a character string, xs:string type

val: xs:long type

Sub-element: N/A

4.14.4 DATE

Properties:

name: a character string, xs:string type

val: xs:date type

Sub-element*:* N/A

4.14.5 TIME

Properties*:*

name: a character string, xs:string type

val: xs:time type

Sub-element*:* N/A

4.14.6 DATETIME

Properties:

name: a character string, xs:string type

val: xs:datetime type

Sub-element: N/A

4.14.7 DURATION

Properties:

name: a character string, xs:string type

val: xs:duration type

Sub-element: N/A

4.14.8 STRING

Properties:

name: a character string, xs:string type

val: xs:string type

Sub-element: N/A

4.14.9 BINARY

Properties:

name: a character string, xs:string type

val: xs:base64Binary type

Sub-element: N/A

4.14.10 BOOL

Properties:

name: a character string, xs:string type

val: xs:boolean type

Sub-element: N/A

4.14.11 COMPOUND

Property:

name: a character string, xs:string type

Sub-element:

arc: ARC type

bezier: BEZIER type

circle: CIRCLE type

cmd: CMD type

rgb: COLOR_RGB type

doc: DOC type

docbase: DOCBASE type

docset: DOCSET type

ellipse: ELLIPSE type

embedfont: EMBEDFONT type

fontlist: FONTLIST type

fontmap: FONTMAP type

image: IMAGE type

layer: LAYER type

line: LINE type

matrix: MATRIX type

meta: META type

metalist: METALIST type

page: PAGE type

path: PATH type

rect: RECT type

roundrect: ROUNDRECT type

subpath: SUBPATH type

text: TEXT type

objstream: OBJSTREAM type

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

4.15 Data Ranges

The following are the general rules for data ranges:

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

5. Conformance

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

Use of the word “shall” indicates required behavior.

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

5.1.1 DCMS Conformance

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

5.1.2 Application Conformance

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

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

Annex A.UOML XML Schema

This annex is informative.

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

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.**.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0"
targetNamespace="urn:oasis:names:tc:uoml:xmlns:uoml:1.0"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
      <xs:complexType name="ARC">
            <xs:annotation>
                  <xs:documentation>arc</xs:documentation>
            </xs:annotation>
            <xs:attribute name="clockwise" type="xs:boolean" use="required"/>
            <xs:attribute name="start" type="xs:string" use="required"/>
            <xs:attribute name="end" type="xs:string" use="required"/>
            <xs:attribute name="center" type="xs:string" use="required"/>
            <xs:attribute name="angle" type="xs:float" use="required"/>
      </xs:complexType>
      <xs:complexType name="BEZIER">
            <xs:annotation>
                  <xs:documentation>bezier curve</xs:documentation>
            </xs:annotation>
            <xs:attribute name="start" type="xs:string" use="required"/>
            <xs:attribute name="ctrl" type="xs:string" use="required"/>
            <xs:attribute name="ctrl2" type="xs:string" use="optional"/>
            <xs:attribute name="end" type="xs:string" use="required"/>
      </xs:complexType>
      <xs:complexType name="CIRCLE">
            <xs:annotation>
                  <xs:documentation>circle</xs:documentation>
            </xs:annotation>
            <xs:attribute name="radius" type="xs:int" use="required"/>
            <xs:attribute name="center" type="xs:string" use="required"/>
      </xs:complexType>
      <xs:complexType name="LINE">
            <xs:annotation>
                  <xs:documentation>line</xs:documentation>
            </xs:annotation>
```

```
			<xs:attribute name="start" type="xs:string" use="required"/>
			<xs:attribute name="end" type="xs:string" use="required"/>
		</xs:complexType>
		<xs:complexType name="RECT">
			<xs:annotation>
				<xs:documentation>rect</xs:documentation>
			</xs:annotation>
			<xs:attribute name="tl" type="xs:string" use="required"/>
			<xs:attribute name="br" type="xs:string" use="required"/>
		</xs:complexType>
		<xs:complexType name="ELLIPSE">
			<xs:annotation>
				<xs:documentation>ellipse</xs:documentation>
			</xs:annotation>
			<xs:attribute name="xr" type="xs:int" use="required"/>
			<xs:attribute name="yr" type="xs:int" use="required"/>
			<xs:attribute name="center" type="xs:string" use="required"/>
			<xs:attribute name="angle" type="xs:float" use="required"/>
		</xs:complexType>
		<xs:complexType name="ROUNDRECT">
			<xs:annotation>
				<xs:documentation>roundrect</xs:documentation>
			</xs:annotation>
			<xs:attribute name="xr" type="xs:int" use="required"/>
			<xs:attribute name="yr" type="xs:int" use="required"/>
			<xs:attribute name="tl" type="xs:string" use="required"/>
			<xs:attribute name="br" type="xs:string" use="required"/>
		</xs:complexType>
		<xs:complexType name="META">
			<xs:annotation>
				<xs:documentation>metadata</xs:documentation>
			</xs:annotation>
			<xs:attribute name="key" type="xs:string" use="required"/>
			<xs:attribute name="val" type="xs:string" use="required"/>
		</xs:complexType>
		<xs:complexType name="METALIST">
			<xs:annotation>
				<xs:documentation>metadata list</xs:documentation>
			</xs:annotation>
			<xs:sequence>
				<xs:element name="meta" type="uoml:META" minOccurs="0"
	maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
		<xs:complexType name="CMD">
			<xs:annotation>
				<xs:documentation>cmd</xs:documentation>
			</xs:annotation>
			<xs:sequence minOccurs="0">
```

```
                        <xs:choice>
                              <xs:element name="cliparea" type="uoml:PATH"/>
                              <xs:element name="matrix" type="uoml:MATRIX"/>
                              <xs:element name="rgb" type="uoml:COLOR_RGB"/>
                        </xs:choice>
                  </xs:sequence>
                  <xs:attribute name="name" type="uoml:CMDNAME" use="required"/>
                  <xs:attribute name="v1" type="xs:anySimpleType"/>
                  <xs:attribute name="v2" type="xs:anySimpleType"/>
            </xs:complexType>
            <xs:complexType name="MATRIX">
                  <xs:annotation>
                        <xs:documentation>matrix</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="f11" type="xs:float" use="required"/>
                  <xs:attribute name="f12" type="xs:float" use="required"/>
                  <xs:attribute name="f21" type="xs:float" use="required"/>
                  <xs:attribute name="f22" type="xs:float" use="required"/>
                  <xs:attribute name="f31" type="xs:float" use="required"/>
                  <xs:attribute name="f32" type="xs:float" use="required"/>
            </xs:complexType>
            <xs:complexType name="SUBPATH">
                  <xs:annotation>
                        <xs:documentation>subpath</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="data" type="xs:string" use="required"/>
            </xs:complexType>
            <xs:complexType name="PATH">
                  <xs:annotation>
                        <xs:documentation>path</xs:documentation>
                  </xs:annotation>
                  <xs:sequence>
                        <xs:choice minOccurs="0" maxOccurs="unbounded">
                              <xs:element name="subpath" type="uoml:SUBPATH"/>
                              <xs:element name="rect" type="uoml:RECT"/>
                              <xs:element name="circle" type="uoml:CIRCLE"/>
                              <xs:element name="ellipse" type="uoml:ELLIPSE"/>
                              <xs:element name="roundrect" type="uoml:ROUNDRECT"/>
                        </xs:choice>
                  </xs:sequence>
            </xs:complexType>
            <xs:complexType name="COLOR_RGB">
                  <xs:annotation>
                        <xs:documentation>rgb color</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="r" type="xs:short" use="required"/>
                  <xs:attribute name="g" type="xs:short" use="required"/>
                  <xs:attribute name="b" type="xs:short" use="required"/>
                  <xs:attribute name="a" type="xs:short" use="optional"/>
```

```
	</xs:complexType>
	<xs:complexType name="EMBEDFONT">
		<xs:annotation>
			<xs:documentation>embedded font</xs:documentation>
		</xs:annotation>
		<xs:simpleContent>
			<xs:extension base="xs:base64Binary">

			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:complexType name="FONTMAP">
		<xs:annotation>
			<xs:documentation>font mapping</xs:documentation>
		</xs:annotation>
		<xs:attribute name="name" type="xs:string" use="required"/>
		<xs:attribute name="no" type="xs:int" use="required"/>
	</xs:complexType>
	<xs:complexType name="FONTLIST">
		<xs:annotation>
			<xs:documentation>font list</xs:documentation>
		</xs:annotation>
	</xs:complexType>
	<xs:complexType name="IMAGE">
		<xs:annotation>
			<xs:documentation>image</xs:documentation>
		</xs:annotation>
		<xs:simpleContent>
			<xs:extension base="xs:base64Binary">
			  <xs:attribute name="tl" type="xs:string" use="required"/>
			  <xs:attribute name="br" type="xs:string" use="required"/>
			  <xs:attribute name="type" type="xs:string" use="required"/>
			  <xs:attribute name="path" type="xs:string" use="optional"/>
			</xs:extension>
		</xs:simpleContent>

	</xs:complexType>
	<xs:complexType name="TEXT">
		<xs:annotation>
			<xs:documentation>text</xs:documentation>
		</xs:annotation>
		<xs:attribute name="origin" type="xs:string" use="required"/>
		<xs:attribute name="encode" type="xs:string" use="required"/>
		<xs:attribute name="text" type="xs:string" use="required"/>
		<xs:attribute name="spaces" type="xs:string" use="optional"/>
	</xs:complexType>
	<xs:simpleType name="CMDNAME">
		<xs:annotation>
			<xs:documentation>command names</xs:documentation>
		</xs:annotation>
```

```
            <xs:restriction base="xs:string">
                  <xs:enumeration value="COLOR_LINE"/>
                  <xs:enumeration value="COLOR_FILL"/>
                  <xs:enumeration value="COLOR_TEXT"/>
                  <xs:enumeration value="COLOR_SHADOW"/>
                  <xs:enumeration value="COLOR_OUTLINE"/>
                  <xs:enumeration value="LINE_WIDTH"/>
                  <xs:enumeration value="LINE_JOIN"/>
                  <xs:enumeration value="LINE_CAP"/>
                  <xs:enumeration value="MITER_LIMIT"/>
                  <xs:enumeration value="FILL_RULE"/>
                  <xs:enumeration value="RENDER_MODE"/>
                  <xs:enumeration value="RASTER_OP"/>
                  <xs:enumeration value="TEXT_DIR"/>
                  <xs:enumeration value="CHAR_DIR"/>
                  <xs:enumeration value="CHAR_ROTATE"/>
                  <xs:enumeration value="CHAR_SLANT"/>
                  <xs:enumeration value="CHAR_SIZE"/>
                  <xs:enumeration value="CHAR_WEIGHT"/>
                  <xs:enumeration value="CHAR_STYLE"/>
                  <xs:enumeration value="TEXT_MATRIX"/>
                  <xs:enumeration value="IMAGE_MATRIX"/>
                  <xs:enumeration value="GRAPH_MATRIX"/>
                  <xs:enumeration value="EXT_MATRIX"/>
                  <xs:enumeration value="PUSH_GS"/>
                  <xs:enumeration value="POP_GS"/>
                  <xs:enumeration value="SHADOW_WIDTH"/>
                  <xs:enumeration value="SHADOW_DIR"/>
                  <xs:enumeration value="SHADOW_LEN"/>
                  <xs:enumeration value="SHADOW_NEG"/>
                  <xs:enumeration value="SHADOW_ATL"/>
                  <xs:enumeration value="CLIP_AREA"/>
                  <xs:enumeration value="FONT"/>
                  <xs:enumeration value="OUTLINE_BORDER"/>
                  <xs:enumeration value="OUTLINE_WIDTH"/>
                  <xs:enumeration value="HOLLOW_BORDER"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name="LINECAP">
            <xs:annotation>
                  <xs:documentation>line cap style</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                  <xs:enumeration value="END_BUTT"/>
                  <xs:enumeration value="END_SQUARE"/>
                  <xs:enumeration value="END_ROUND"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name="JOINCAP">
```

```
            <xs:annotation>
                  <xs:documentation>line join style</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                  <xs:enumeration value="JOIN_MITER"/>
                  <xs:enumeration value="JOIN_BEVEL"/>
                  <xs:enumeration value="JOIN_ROUND"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name="FILLRULE">
            <xs:annotation>
                  <xs:documentation>fill rule</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                  <xs:enumeration value="RULE_EVENODD"/>
                  <xs:enumeration value="RULE_WINDING"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name="ROP">
            <xs:annotation>
                  <xs:documentation>rop operation</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                  <xs:enumeration value="ROP_COPY"/>
                  <xs:enumeration value="ROP_N_COPY"/>
                  <xs:enumeration value="ROP_RESET"/>
                  <xs:enumeration value="ROP_SET"/>
                  <xs:enumeration value="ROP_NOP"/>
                  <xs:enumeration value="ROP_REV"/>
                  <xs:enumeration value="ROP_AND"/>
                  <xs:enumeration value="ROP_AND_N"/>
                  <xs:enumeration value="ROP_N_AND"/>
                  <xs:enumeration value="ROP_N_AND_N"/>
                  <xs:enumeration value="ROP_OR"/>
                  <xs:enumeration value="ROP_OR_N"/>
                  <xs:enumeration value="ROP_N_OR"/>
                  <xs:enumeration value="ROP_N_OR_N"/>
                  <xs:enumeration value="ROP_XOR"/>
                  <xs:enumeration value="ROP_EOR"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name="CHARTXTDIR">
            <xs:annotation>
                  <xs:documentation>text or char direction</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                  <xs:enumeration value="HEAD_LEFT"/>
                  <xs:enumeration value="HEAD_RIGHT"/>
                  <xs:enumeration value="HEAD_TOP"/>
            <xs:enumeration value="HEAD_BOTTOM"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="SHADOWDIR">
        <xs:annotation>
            <xs:documentation>shadow direction</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="SHADOW_LT"/>
            <xs:enumeration value="SHADOW_LB"/>
            <xs:enumeration value="SHADOW_RT"/>
            <xs:enumeration value="SHADOW_RB"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="OBJSTREAM">
        <xs:annotation>
            <xs:documentation>object stream</xs:documentation>
        </xs:annotation>
    </xs:complexType>
    <xs:complexType name="LAYER">
        <xs:annotation>
            <xs:documentation>layer</xs:documentation>
        </xs:annotation>
    </xs:complexType>
    <xs:complexType name="PAGE">
        <xs:annotation>
            <xs:documentation>page</xs:documentation>
        </xs:annotation>
        <xs:attribute name="width" type="xs:float" use="required"/>
        <xs:attribute name="height" type="xs:float" use="required"/>
        <xs:attribute name="resolution" type="xs:int" use="required"/>
    </xs:complexType>
    <xs:complexType name="DOC">
        <xs:annotation>
            <xs:documentation>doc</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="metainfo" type="uoml:METALIST"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:complexType name="DOCSET">
        <xs:annotation>
            <xs:documentation>doc set</xs:documentation>
        </xs:annotation>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:complexType name="DOCBASE">
        <xs:annotation>
```

```
                        <xs:documentation>doc base</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="name" type="xs:string" use="required"/>
                  <xs:attribute name="path" type="xs:string" use="required"/>
            </xs:complexType>
            <xs:element name="CLOSE">
                  <xs:complexType>
                        <xs:attribute name="handle" type="xs:string" use="optional"/>
                  </xs:complexType>
            </xs:element>
            <xs:element name="DELETE">
                  <xs:complexType>
                        <xs:attribute name="handle" type="xs:string" use="optional"/>
                  </xs:complexType>
            </xs:element>
            <xs:element name="INSERT">
                  <xs:complexType>
                        <xs:choice>
                              <xs:element name="xobj" type="uoml:COMPOUND"/>
                        </xs:choice>
                        <xs:attribute name="handle" type="xs:string"/>
                        <xs:attribute name="pos" type="xs:int"/>
                  </xs:complexType>
            </xs:element>
            <xs:element name="GET">
                  <xs:complexType>
                        <xs:choice>
                              <xs:element name="disp_conf">
                                    <xs:complexType>
                                          <xs:sequence>
                                                <xs:element name="clip" type="uoml:PATH"
      minOccurs="0"/>
                                          </xs:sequence>
                                          <xs:attribute name="end_layer" type="xs:int"/>
                                          <xs:attribute name="resolution"
      type="xs:int"/>
                                          <xs:attribute name="format" type="xs:string"/>
                                          <xs:attribute name="output" type="xs:string"
      use="required"/>
                                          <xs:attribute name="addr" type="xs:string"
      use="required"/>
                                    </xs:complexType>
                              </xs:element>
                              <xs:element name="pos">
                                    <xs:complexType>
                                          <xs:attribute name="val" type="xs:int"
      use="required"/>
                                    </xs:complexType>
                              </xs:element>
```

```
                <xs:element name="property">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string"
use="required"/>
                    </xs:complexType>
                </xs:element>
            </xs:choice>
            <xs:attribute name="usage" type="xs:string" use="required"/>
            <xs:attribute name="handle" type="xs:string"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="SET">
        <xs:complexType>
            <xs:choice>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="intVal" type="uoml:INT"/>
                    <xs:element name="floatVal" type="uoml:DOUBLE"/>
                    <xs:element name="timeVal" type="uoml:TIME"/>
                    <xs:element name="dateVal" type="uoml:DATE"/>
                    <xs:element name="dateTimeVal"
type="uoml:DATETIME"/>
                    <xs:element name="durationVal"
type="uoml:DURATION"/>
                    <xs:element name="stringVal" type="uoml:STRING"/>
                    <xs:element name="binaryVal" type="uoml:BINARY"/>
                    <xs:element name="compoundVal"
type="uoml:COMPOUND"/>
                    <xs:element name="boolVal" type="uoml:BOOL"/>
                </xs:choice>
            </xs:choice>
            <xs:attribute name="handle" type="xs:string"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="USE">
        <xs:complexType>
            <xs:attribute name="handle" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="OPEN">
        <xs:complexType>
            <xs:attribute name="create" type="xs:boolean" default="true"/>
            <xs:attribute name="del_exist" type="xs:boolean"
default="false"/>
            <xs:attribute name="path" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="SYSTEM">
        <xs:complexType>
            <xs:choice>
```

```
                        <xs:element name="flush">
                              <xs:complexType>
                                    <xs:attribute name="handle"/>
                                    <xs:attribute name="path"/>
                              </xs:complexType>
                        </xs:element>
                  </xs:choice>
            </xs:complexType>
      </xs:element>
      <xs:element name="RET">
            <xs:complexType>
                  <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element name="intVal" type="uoml:INT"/>
                        <xs:element name="floatVal" type="uoml:DOUBLE"/>
                        <xs:element name="timeVal" type="uoml:TIME"/>
                        <xs:element name="dateVal" type="uoml:DATE"/>
                        <xs:element name="dateTimeVal" type="uoml:DATETIME"/>
                        <xs:element name="durationVal" type="uoml:DURATION"/>
                        <xs:element name="stringVal" type="uoml:STRING"/>
                        <xs:element name="binaryVal" type="uoml:BINARY"/>
                        <xs:element name="compoundVal" type="uoml:COMPOUND"/>
                        <xs:element name="boolVal" type="uoml:BOOL"/>
                        <xs:element name="longVal" type="uoml:LONG"/>
                  </xs:choice>
            </xs:complexType>
      </xs:element>
      <xs:complexType name="COMPOUND">
            <xs:annotation>
                  <xs:documentation>compound parameter type</xs:documentation>
            </xs:annotation>
            <xs:choice minOccurs="0">
                  <xs:element name="arc" type="uoml:ARC"/>
                  <xs:element name="bezier" type="uoml:BEZIER"/>
                  <xs:element name="circle" type="uoml:CIRCLE"/>
                  <xs:element name="cmd" type="uoml:CMD"/>
                  <xs:element name="rgb" type="uoml:COLOR_RGB"/>
                  <xs:element name="doc" type="uoml:DOC"/>
                  <xs:element name="docbase" type="uoml:DOCBASE"/>
                  <xs:element name="docset" type="uoml:DOCSET"/>
                  <xs:element name="ellipse" type="uoml:ELLIPSE"/>
                  <xs:element name="embedfont" type="uoml:EMBEDFONT"/>
                  <xs:element name="fontlist" type="uoml:FONTLIST"/>
                  <xs:element name="fontmap" type="uoml:FONTMAP"/>
                  <xs:element name="image" type="uoml:IMAGE"/>
                  <xs:element name="layer" type="uoml:LAYER"/>
                  <xs:element name="line" type="uoml:LINE"/>
                  <xs:element name="matrix" type="uoml:MATRIX"/>
                  <xs:element name="meta" type="uoml:META"/>
                  <xs:element name="metalist" type="uoml:METALIST"/>
```

```
                        <xs:element name="page" type="uoml:PAGE"/>
                        <xs:element name="path" type="uoml:PATH"/>
                        <xs:element name="rect" type="uoml:RECT"/>
                        <xs:element name="roundrect" type="uoml:ROUNDRECT"/>
                        <xs:element name="subpath" type="uoml:SUBPATH"/>
                        <xs:element name="text" type="uoml:TEXT"/>
                        <xs:element name="objstream" type="uoml:OBJSTREAM"/>
                  </xs:choice>
                  <xs:attribute name="name" type="xs:string"/>
            </xs:complexType>
            <xs:complexType name="STRING">
                  <xs:annotation>
                        <xs:documentation>string parameter type</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="val" type="xs:string" use="required"/>
                  <xs:attribute name="name" type="xs:string"/>
            </xs:complexType>
            <xs:complexType name="DOUBLE">
                  <xs:annotation>
                        <xs:documentation>double precision float parameter
      type</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="val" type="xs:double" use="required"/>
                  <xs:attribute name="name" type="xs:string"/>
            </xs:complexType>
            <xs:complexType name="DATE">
                  <xs:annotation>
                        <xs:documentation>date parameter type</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="val" type="xs:date" use="required"/>
                  <xs:attribute name="name" type="xs:string"/>
            </xs:complexType>
            <xs:complexType name="DATETIME">
                  <xs:annotation>
                        <xs:documentation>date and time parameter
      type</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="val" type="xs:dateTime" use="required"/>
                  <xs:attribute name="name" type="xs:string"/>
            </xs:complexType>
            <xs:complexType name="TIME">
                  <xs:annotation>
                        <xs:documentation>time parameter type</xs:documentation>
                  </xs:annotation>
                  <xs:attribute name="val" type="xs:time" use="required"/>
                  <xs:attribute name="name" type="xs:string"/>
            </xs:complexType>
            <xs:complexType name="DURATION">
                  <xs:annotation>
            <xs:documentation>duration parameter type</xs:documentation>
        </xs:annotation>
        <xs:attribute name="val" type="xs:duration" use="required"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="BINARY">
        <xs:annotation>
            <xs:documentation>binary parameter type</xs:documentation>
        </xs:annotation>
        <xs:attribute name="val" type="xs:base64Binary" use="required"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="INT">
        <xs:annotation>
            <xs:documentation>integer parameter type</xs:documentation>
        </xs:annotation>
        <xs:attribute name="val" type="xs:int" use="required"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="BOOL">
        <xs:annotation>
            <xs:documentation>boolean parameter type</xs:documentation>
        </xs:annotation>
        <xs:attribute name="val" type="xs:boolean" use="required"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="LONG">
        <xs:annotation>
            <xs:documentation>long parameter type</xs:documentation>
        </xs:annotation>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="val" type="xs:long" use="required"/>
    </xs:complexType>
    <xs:simpleType name="CHARSTYLE">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SHADOW"/>
            <xs:enumeration value="HOLLOW"/>
            <xs:enumeration value="OUTLINE"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

End of informative text.

Annex B. Detailed UOML Examples

This annex is informative.

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

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

Example 1: open a docbase

Instructions sent from application to DCMS

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

Instructions returned from DCMS to application

```
<!-- the string value “docbase001” is the opened docbase’s handle for later use -->

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

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

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

</uoml:RET>
```

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

Instructions sent from application to DCMS

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

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

  <pos val="0"/>

</uoml:GET>
```

Instructions returned from DCMS to application

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

  <boolVal name="SUCCESS" val="true"/>
  <stringVal name="handle" val="docset001"/>
</uoml:RET>
```

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

Instructions sent from application to DCMS

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

Instructions returned from DCMS to application

```
<!-- the return value of 3 indicates the root docset has 3 sub-objects -->
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
  <intVal name="sub_count" val="3"/>
</uoml:RET>
```

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

Instructions sent from application to DCMS

```
<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docset001"
usage="GET_SUB">
  <pos val="2"/>
</uoml:GET>
```

Instructions returned from DCMS to application

```
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
  <stringVal name="handle" val="doc001"/>
</uoml:RET>
```

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

Instructions sent from application to DCMS

```
<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP"
handle="doc001">
  <property name=""/>
```

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

Instructions returned from DCMS to application

```
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
  <stringVal name="" val="DOC"/>
</uoml:RET>
```

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

Instructions sent from application to DCMS

```
<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP"
handle="doc001">
  <property name="metainfo"/>
</uoml:GET>
```

Instructions returned from DCMS to application

```
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
  <compoundVal name="metainfo">
       <metalist>
              <meta key= "title" val="UOML Part I"/>
              <meta key="author" val="UOML TC"/>
       </metalist>
  </compoundVal>
</uoml:RET>
```

Example 7: get page bitmap of a page

Instructions sent from application to DCMS

```
<!-- the page object's handle is supposed to have already obtained of value "page001" in
prior instructions(using GET) -->
<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PAGE_BMP"
handle="page001">
```

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

Instructions returned from DCMS to application

```
<!-- the bmp format of page bitmap data has been saved in the file c:\test.bmp as requested
-->
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
</uoml:RET>
```

Example 8 : get first layer of a page

Instructions sent from application to DCMS

```
<!-- the page object's handle is supposed to have already obtained of value "page001" in
prior instructions(using GET) -->
<!-- since page has only layer objects as its sub-objects, get sub-objects is the same to
get layer objects -->
<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="page001"
usage="GET_SUB">
  <pos val="0"/>
</uoml:GET>
```

Instructions returned from DCMS to application

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

Example 9: set a text object as the current object

```
Instructions send from application to DCMS

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

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

Instructions returned from DCMS to application

<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
</uoml:RET>
```

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

```
Instructions send from application to DCMS

<uoml:GET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" usage="GET_PROP">
  <property name="spaces"/>
</uoml:GET>

Instructions returned from DCMS to application

<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
  <stringVal name="spaces" val="50,55"/>
</uoml:RET>
```

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

```
Instructions send from application to DCMS

<uoml:INSERT xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0" handle="docset001">
  <xobj>
    <doc name="UOML part II">
      <metainfo>
        <meta key="author" val="alex"/>
      </metainfo>
    </doc>
  </xobj>
</uoml:INSERT>
```

Instructions returned from DCMS to application

```
<!-- the handle of the inserted document is returned for later use  -->
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
  <stringVal name="handle" val="doc002"/>
</uoml:RET>
```

Example 12: delete the document inserted in the example above

Instructions send from application to DCMS

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

Instructions returned from DCMS to application

```
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
</uoml:RET>
```

Example 13: use SYSTEM to save a docbase

Instructions send from application to DCMS

```
<uoml:SYSTEM xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <flush path="c:\test.sep"/>
</uoml:SYSTEM>
<!-- instructions returned from DCMS to application -->
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
</uoml:RET>
```

Example 14: close the docbase (following example 1)

Instructions send from application to DCMS

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

instructions returned from DCMS to application

```
<uoml:RET xmlns:uoml="urn:oasis:names:tc:uoml:xmlns:uoml:1.0">
  <boolVal name="SUCCESS" val="true"/>
</uoml:RET>
```

End of informative text.

Annex C.RELAX NG Representation of the UOML XML Schema

This annex is informative.

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

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

start =
  (notAllowed
   | element ns1:OPEN {
       attribute path { xsd:string },
       attribute del_exist { xsd:boolean }?,
       attribute create { xsd:boolean }?
     })
  | (notAllowed
     | element ns1:RET {
         (element intVal { INT }
          | element floatVal { DOUBLE }
          | element timeVal { TIME }
          | element dateVal { DATE }
          | element dateTimeVal { DATETIME }
          | element durationVal { DURATION }
          | element stringVal { STRING }
          | element binaryVal { BINARY }
          | element compoundVal { COMPOUND }
          | element boolVal { BOOL }
          | element longVal {
              attribute val { xsd:long },
              attribute name { xsd:string }?
            })*
       })
  | (notAllowed
     | element ns1:SET {
         attribute handle { xsd:string }?,
         (element intVal { INT }
          | element floatVal { DOUBLE }
          | element timeVal { TIME }
          | element dateVal { DATE }
          | element dateTimeVal { DATETIME }
          | element durationVal { DURATION }
          | element stringVal { STRING }
          | element binaryVal { BINARY }
          | element compoundVal { COMPOUND }
          | element boolVal { BOOL })*
      })
  | (notAllowed
     | element ns1:GET {
         attribute handle { xsd:string }?,
         attribute usage { xsd:string },
         (element disp_conf {
            attribute addr { xsd:string },
            attribute output { xsd:string },
            attribute format { xsd:string }?,
            attribute resolution { xsd:int }?,
            attribute end_layer { xsd:int }?,
            element clip { PATH }?
          }
          | element pos {
              attribute val { xsd:int }
            }
          | element property {
              attribute name { xsd:string }
            })
      })
  | (notAllowed
     | element ns1:DELETE {
         attribute handle { xsd:string }?
      })
  | (notAllowed
     | element ns1:USE {
         attribute handle { xsd:string }
      })
  | (notAllowed
     | element ns1:INSERT {
         attribute pos { xsd:int }?,
         attribute handle { xsd:string }?,
         element xobj { COMPOUND }
      })
  | (notAllowed
     | element ns1:SYSTEM {
         element flush {
           attribute path { text }?,
           attribute handle { text }?
         }
      })
  | (notAllowed
     | element ns1:CLOSE {
         attribute handle { xsd:string }?
      })
COMPOUND =
  (attribute name { xsd:string }?,
```

Copyright© OASIS® 1993-2010. All Rights Reserved.

```
    ((notAllowed
      | element arc {
          attribute angle { xsd:float },
          attribute center { xsd:string },
          attribute end { xsd:string },
          attribute start { xsd:string },
          attribute clockwise { xsd:boolean }
        })
     | (notAllowed
        | element bezier {
            attribute end { xsd:string },
            attribute ctrl2 { xsd:string }?,
            attribute ctrl { xsd:string },
            attribute start { xsd:string }
          })
     | (notAllowed
        | element circle { CIRCLE })
     | (notAllowed
        | element cmd {
            attribute v2 {
              text
              # <data type="anySimpleType"/>

            }?,
            attribute v1 {
              text
              # <data type="anySimpleType"/>

            }?,
            attribute name {
              xsd:string "CHAR_WEIGHT"
              | xsd:string "CLIP_AREA"
              | xsd:string "COLOR_FILL"
              | xsd:string "CHAR_SIZE"
              | xsd:string "LINE_CAP"
              | xsd:string "SHADOW_LEN"
              | xsd:string "CHAR_STYLE"
              | xsd:string "RENDER_MODE"
              | xsd:string "CHAR_SLANT"
              | xsd:string "COLOR_LINE"
              | xsd:string "TEXT_DIR"
              | xsd:string "COLOR_TEXT"
              | xsd:string "GRAPH_MATRIX"
              | xsd:string "HOLLOW_BORDER"
              | xsd:string "POP_GS"
              | xsd:string "PUSH_GS"
              | xsd:string "LINE_WIDTH"
              | xsd:string "CHAR_DIR"
              | xsd:string "OUTLINE_WIDTH"
```

```
                 | xsd:string "FILL_RULE"
                 | xsd:string "EXT_MATRIX"
                 | xsd:string "SHADOW_WIDTH"
                 | xsd:string "RASTER_OP"
                 | xsd:string "TEXT_MATRIX"
                 | xsd:string "LINE_JOIN"
                 | xsd:string "SHADOW_NEG"
                 | xsd:string "SHADOW_ATL"
                 | xsd:string "CHAR_ROTATE"
                 | xsd:string "MITER_LIMIT"
                 | xsd:string "COLOR_OUTLINE"
                 | xsd:string "FONT"
                 | xsd:string "IMAGE_MATRIX"
                 | xsd:string "SHADOW_DIR"
                 | xsd:string "OUTLINE_BORDER"
                 | xsd:string "COLOR_SHADOW"
               },
               (element cliparea { PATH }
                | element matrix { MATRIX }
                | element rgb { COLOR_RGB })?
             })
        | (notAllowed
           | element rgb { COLOR_RGB })
        | (notAllowed
           | element doc {
               attribute name { xsd:string },
               element metainfo { METALIST }
             })
        | (notAllowed
           | element docbase {
               attribute path { xsd:string },
               attribute name { xsd:string }
             })
        | (notAllowed
           | element docset {
               attribute name { xsd:string }
             })
        | (notAllowed
           | element ellipse { ELLIPSE })
        | (notAllowed
           | element embedfont { xsd:base64Binary })
        | (notAllowed
           | element fontlist { empty })
        | (notAllowed
           | element fontmap {
               attribute no { xsd:int },
               attribute name { xsd:string }
             })
        | (notAllowed
           | element image {
               attribute tl { xsd:string },
               attribute br { xsd:string },
               attribute type { xsd:string },
               attribute path { xsd:string }?,
               xsd:base64Binary
             })
        | (notAllowed
           | element layer { empty })
        | (notAllowed
           | element line {
               attribute end { xsd:string },
               attribute start { xsd:string }
             })
        | (notAllowed
           | element matrix { MATRIX })
        | (notAllowed
           | element meta { META })
        | (notAllowed
           | element metalist { METALIST })
        | (notAllowed
           | element page {
               attribute resolution { xsd:int },
               attribute height { xsd:float },
               attribute width { xsd:float }
             })
        | (notAllowed
           | element path { PATH })
        | (notAllowed
           | element rect { RECT })
        | (notAllowed
           | element roundrect { ROUNDRECT })
        | (notAllowed
           | element subpath { SUBPATH })
        | (notAllowed
           | element text {
               attribute spaces { xsd:string }?,
               attribute text { xsd:string },
               attribute encode { xsd:string },
               attribute origin { xsd:string }
             })
        | (notAllowed
           | element objstream { empty }))?),
     empty
   PATH =
     ((notAllowed
        | element subpath { SUBPATH })
       | (notAllowed
          | element rect { RECT })
       | (notAllowed
          | element circle { CIRCLE })
       | (notAllowed
          | element ellipse { ELLIPSE })
       | (notAllowed
          | element roundrect { ROUNDRECT }))*,
     empty
   METALIST =
     (notAllowed
       | element meta { META })*,
     empty
   COLOR_RGB =
     (attribute a { xsd:short }?,
      attribute b { xsd:short },
      attribute g { xsd:short },
      attribute r { xsd:short }),
     empty
   TIME =
     (attribute name { xsd:string }?,
      attribute val { xsd:time }),
     empty
   ELLIPSE =
     (attribute angle { xsd:float },
      attribute center { xsd:string },
      attribute yr { xsd:int },
      attribute xr { xsd:int }),
     empty
   SUBPATH =
     attribute data { xsd:string },
     empty
   INT =
     (attribute name { xsd:string }?,
      attribute val { xsd:int }),
     empty
   DURATION =
     (attribute name { xsd:string }?,
      attribute val { xsd:duration }),
     empty
   ROUNDRECT =
     (attribute br { xsd:string },
      attribute tl { xsd:string },
      attribute yr { xsd:int },
      attribute xr { xsd:int }),
     empty
   DATE =
     (attribute name { xsd:string }?,
      attribute val { xsd:date }),
     empty
   BINARY =
     (attribute name { xsd:string }?,
      attribute val { xsd:base64Binary }),
     empty
   STRING =
     (attribute name { xsd:string }?,
      attribute val { xsd:string }),
     empty
   DOUBLE =
     (attribute name { xsd:string }?,
      attribute val { xsd:double }),
     empty
   BOOL =
     (attribute name { xsd:string }?,
      attribute val { xsd:boolean }),
     empty
   CIRCLE =
     (attribute center { xsd:string },
      attribute radius { xsd:int }),
     empty
   META =
     (attribute val { xsd:string },
      attribute key { xsd:string }),
     empty
   MATRIX =
     (attribute f32 { xsd:float },
      attribute f31 { xsd:float },
      attribute f22 { xsd:float },
      attribute f21 { xsd:float },
      attribute f12 { xsd:float },
      attribute f11 { xsd:float }),
     empty
   RECT =
     (attribute br { xsd:string },
      attribute tl { xsd:string }),
     empty
   DATETIME =
     (attribute name { xsd:string }?,
      attribute val { xsd:dateTime }),
     empty
```

End of informative text.

Annex D.Acknowledgements

This annex is informative.

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

Participants:

Alex Wang, Sursen Corporation

Xu Guo, Sursen Corporation

Ningsheng Liu, Sursen Corporation

Allison Shi, Sursen Corporation

Stephen Green, Individual

Kaihong Zou, Sursen Corporation

Pine Zhang, UOML Alliance

Mendy Liu, UOML Alliance

Joel Marcey, Sursen Corporation

Andy Li, Changfeng Open Standards Platform Software Alliance

Charles H. Schulz, Ars Aperta

Lin Cheng, Beijing Redflag CH2000 Software Co. Ltd.

Liwei Wang, Sursen Corporation

End of informative text.