UOML (Unstructured Operation Markup Language) Part 1 Version 1.0 Draft Errata

Committee Draft 01

23 September 2010

Specification URIs:
This Version:

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

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

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/uoml-part1-v1.0-errata.doc

http://docs.oasis-open.org/uoml-x/v1.0/errata/uoml-part1-v1.0-errata.pdf

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)

Bo Yan, Sursen Corporation <yanbo@sursen.com> (until September 2007)

Editor(s):

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

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

Joel Marcey, Sursen Corporation <joel@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:

This specification is related to: UOML (Unstructured Operation Markup Language) Part 1 v1.0

Declared XML Namespace(s):

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

Abstract:

This document lists errata for the OASIS UOML (Unstructured Operation Markup Language) Version 1.0 Standard product by the UOML-X (Extended) Technical Committee (TC).

Status:

This document was last revised by the OASIS Unstructured Operation Markup Language eXtended (UOML-X) 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

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 names "OASIS", is the 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

List of Changes

1. Summary

There is a proposal to edit the UOML specification substantially. The specification will be reformatted, restructured and thoroughly edited. New normative wording will be added for preciseness (however, breaking changes have been minimized). This errata document will be extensive in nature because the resulting update to the specification will be quite different from the 1.0 standard, especially from a clause arrangement and formatting perspective. However, there are no breaking changes to implementers amongst all of these changes. The flow of this errata document attempts to be in order of the original specification while showing the new structure of any updated specification.

Comment [JM1]: For any errata that has not been covered by a specific comment, or rejected explicitly, DE-29 covers the general improvement of the specification.

The acceptance of GE-07 allows for editorial changes to be made for clarity.

GB-31 was accepted with respect to addition instruction wording for error clarification, etc.

NL-04 allows for restructure and reformatting for better quality. As well as strengthening technical details.

2. Proposed Specification Restructure.

Instead of four primary clauses, there will now be five primary clauses (not including Annexes).

Comment [JM2]: The entire specification restructure is covered by the accepted FR-02 comment.

Specifically too, GB-40 wanted some restructuring.

Some of the front matter made the document more ISO friendly. NO-03 accepts this change.

Before

1. Introduction
2. UOML Document Structure
3. UOML Instructions
4. Conformance
5. Appendix A

After

1. Introduction
2. Abstract Document Model
3. UOML Instructions
4. UOML Objects
5. Conformance
6. Appendices

3. Add Introductory Text for the Introduction Clause.

Proposal to provide some context to the UOML specification within the introductory clause. (current clause 1; proposed new clause 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

4. Editorially Modify the Terminology Clause and add new terms.

Enhanced the Terminology introductory text. Alphabetized the terminology list. Added new terms for *application software*, *DCMS*, *graphics state*, *graphics state stack*, *layer*, *object*, *page bitmap*, *stack*, *sub-element*. Editorially modified the definitions for *command object*, *DCMS*, *docbase*, *Docbase Management System*, *docset*, *graphics object*, *layer*, *object stream*, *Path*, *sub-object*, *UOML*. (current clause 1.1; proposed new clause 1.1)

Before

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Keywords:

Comment [JM3]: NO-04 accepted this.

UOML: short for "Unstructured Operation Markup Language".

Docbase: comes from "Database", means document base, is the container of mass documents; it is the root level of the UOML document structure.

DoCbase Management System: the software which implements the function defined by UOML, short as DCMS.

Docset: a set of documents, like directory in file system.

Layer: a page is composed of one or more layers, each layer has the same size as the page, and the visual appearance of the page is added up by these layers.

Path: refers to the open or closed region collection, which consists of one or multiple line/curve segment(s), its first letter should always be uppercase. In this document, we also use 'path' (all lowercase) to refer to filename, location of Docbase or image file, it is different from 'Path'.

Graphics Object: refers to the objects that could make render engine to draw, it is used to describe the appearance of a page. It includes: text, image, Path, etc.

Command Object: uses for modifying the current graphics state that holds current graphics control parameters, such as text size, typeface and color.

Object Stream: a sequence of graphics objects and command objects.

Sub-object: in a tree structure, the upper level object is called parent object, and its' connected lower level object is called sub-object. One parent object can connect multiple sub-objects, but one sub-object can only have one parent object. Sub-object is created by INSERT instruction

After

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*].

Comment [JM4]: NO-05 rejected this, but DE-09 accepted this. Also GB-10 accepted this.

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*].

Comment [JM5]: NO-05 rejected this, but DE-09 accepted this. Also GB-10 accepted this.

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

Comment [JM6]: Accepted by GB-23.

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

Comment [JM7]: GB-51 accepted this.

Comment [JM8]: Accepted by GB-29

Comment [JM9]: GB-11 Accepted

Comment [JM10]: GB-56 accepted this.

Comment [JM11]: NO-06 rejected this, but GB-12 accepted it.

Comment [JM12]: GB-29 accepted this change. We added the informative note for further explanation.

Comment [JM13]: NO-06 rejected this, but GB-12 accepted it.

Comment [JM14]: FR-04, but DE12 accepted this. Also GB-22 accepted it.

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".

Comment [JM15]: DE-08, GE-07 and NL-02 accepted the general notion of expanding the Terminology section

5. Add a new Scope clause (Normative).

(proposed new clause 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 other parts of this standard.

Comment [JM16]: JP-01, NO-01, NO-17 rejected the entire clause, but DE-03 accepted this. So going with acceptance. GB-04 accepted this as well. NL-01 also accepted this. NO-02 accepted this too.

6. Add a Notational Conventions Clause (Normative).

(proposed new clause 1.3)

Notational Conventions

The following typographical conventions are used in this OASIS standard:

The first occurrence of a new term is written in italics, as in "normative".

In each definition of a term in §1.1 (Terminology), the term is written in bold, as in "docset".

Comment [JM17]: NO-18 accepted this proposed addition.

Comment [JM18]: NO-01 rejected the entire clause, but DE-03 accepted this. So going with acceptance.

7. Add an Acronyms and Abbreviations Clause (Informative).

(proposed new clause 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

Comment [JM19]: NO-01 rejected the entire clause, but DE-03 accepted this. So going with acceptance.

8. Add a General Description Clause (Normative).

A proposal to add a clause describing the structure of the document and how informative text is described. (proposed new clause 1.5)

General Description

This OASIS standard is divided into the following subdivisions:

1. Front matter (clause summary);
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 form the normative pieces of this Part of this OASIS standard:

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

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

- Introductory text in clause 1
- Subclauses 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]

Comment [JM20]: NO-01 rejected the entire clause, but DE-03 accepted this. So going with acceptance.

9. Enhance and Editorially Modify the Overview Clause.

Added wording about relationship to PDF, future UOML Parts, and made general editorial changes (current clause 1.2; proposed new clause 1.6).

Before

Overview

UOML is interface standard to process unstructured document; it plays the similar role as SQL (Structured Query Language) to structured data. UOML is expressed with standard XML, featuring compatibility and openness

UOML deals with layout-based document and its related information (such as metadata, rights, etc.) Layout-based document is two dimensional, static paging information, i.e. information can be recorded on traditional paper. The software which implements the UOML defined function, is called DCMS, applications can process the document by sending UOML instructions to DCMS.

UOML first defines abstract document model, then operations to the model. Those operations include read/write, edit, display/print, query, security control; it covers the operations which required by all different kinds of application software to process documents. UOML is based on XML description, and is platform-independent, application-independent, programming language-independent, and vendor neutral. This standard will not restrict manufacturers to implement DCMS in their own specific way.

This specification is the 1st part of UOML, which defines the operations used for read/write, edit, and display/print layout-based document.

This specification defines UOML objects and UOML instructions as following.

After

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.

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.

Comment [JM21]: NO-10 accepts this change even though NO-11 rejected it.

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.

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

End of informative text

Comment [JM22]: GB-03, GB-06 rejected this but DE-07 accepted it.

Comment [JM23]: FR-06 rejcted this, but GB-05 accepted it.

Comment [JM24]: NO-01, NO-09 rejected the entire clause but DE-03 accepted this. So going with acceptance.

10. Add items to the Normative References Clause.

Items were added to the normative references clause. Also slight changes were made to existing normative references: *XML 1.0*, *xml-names*, *xmlschema-1*, *xmlschema-2, RFC2119, PNG, JBIG, JPEG, OpenFont.* Editorial edits were made as well. (current clause 1.3; proposed new clause 1.7)

Before

Normative References

[XML1.0] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau , Extensible Markup Language (XML) 1.0 (Third Edition),

http://www.w3.org/TR/2004/REC-xml-20040204, W3C, 2004.

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

[xmlschema-1] W3C XML Schema Definition Language (XSDL) 1.1 Part 1: Structures http://www.w3.org/TR/xmlschema11-1/

[xmlschema-2] Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes Second Edition, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/, W3C, 2004.

[RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[**PNG**] ISO/IEC 15948:2004 Portable Network Graphics (PNG): Functional specification

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29581.

[JBIG] ISO/IEC 11544, Coded representation of picture and audio information -- Progressive bi-level image compression

[JPEG] ISO/IEC 10918, Digital compression and coding of continuous-tone still images

[OpenFont] ISO/IEC 14496-22:2007, "Open Font Format Specification"

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43466

After

Normative References

The following referenced documents are indispensable for the application 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

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

[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

Comment [JM25]: DE-20, GB-16 accepted this.

Comment [JM26]: DE-23 rejected this, but GB-17 accepted it. GB-41 also accepted it.

Comment [JM27]: CS-05 accepted proposal

Comment [JM28]: DE-20 GB-16 Accepted this.

Comment [JM29]: GB-54 accepted this.

Comment [JM30]: GB-39 accepted this.

Comment [JM31]: NO-17 rejected this, but this is kept as a result of CZ-01 changes to the conformance clause. Also DE-10 accepts this as well.

[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/

Comment [JM32]: Cz-02 accepted proposed changes.

11. Slight Editorial Change to the one Non-Normative Reference.

The document title in the non-normative reference has been italicized. (current clause 1.4; proposed new clause 1.8)

Before

[PDF] ISO/IEC FDIS 32000**,** Portable Document Format based on PDF1.7: Part 1

After

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

12. Rename the UOML Document Structure Clause to Abstract Document Model.

The “UOML Document Structure” clause was renamed to “Abstract Document Model”. (current clause 2; proposed new clause 2)

13. Reword Introduction to Abstract Document Model clause.

(current clause 2; proposed new clause 2)

Before

UOML works on abstract document model. The abstract document model can be regarded as a hierarchy of objects, the UOML instructions deal with these objects. This chapter specifies what kinds of objects are included in abstract document model, and also addresses the detailed description of each object.

This chapter covers the following issues:

- Document Architecture: the relationships among DOCBASE, DOCSET and DOC
- Internal Structure of Document: Global Data and Page Data
- Document Global Data: Metadata and Font
- Page Data
- Graphics Objects
- Command Objects
- Default Value of Graphics State

After

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.

14. Rename Document Architecture sub-clause.

The Document Architecture sub-clause in UOML Document Structure was renamed to Overview. (current clause 2.1; proposed new clause 2.1)

15. Uses of docbase, docset and document in paragraph text all become lowercase.

Anywhere in the specification where the term “Docbase”, “Docset” or “Document” is currently used within the specification (except for clause heading titles and the beginning of sentences, or where grammatically appropriate) will be changed to lowercase versions “docbase”, “docset” and “document”. For example, current clause 2.1, line 108:

Before

Documents are organized with Docbase, Docset and Document

After

Documents are organized with docbase, docset and document

16. Reword current Document Architecture sub-clause which is the proposed Overview sub-clause.

(current clause 2.1; proposed new clause 2.1)

Before

Documents are organized with Docbase, Docset and Document. Within one Docbase, it must have one and only one Docset, as the root Docset, which is the collection and entrance for all the documents, similar to the root directory of a file system. As the container for Document, Docset can be embedded, which means it may contain sub-Docset. Therefore, Docbase, Docset and Document can construct a multiple level tree structure, just like the file system.

After

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.

The following clauses provide a description of each object type.

Comment [JM33]: Accepted by GB-10 and GB-25

17. Move Figures 1 and 2 to new proposed sub-clause Overview.

(current clause 2.2; proposed new clause 2.1)

18. Modify Figure 2 for extended consistency and clarity

Comment [JM34]: Accepted by CZ-03 and DE-04

Before

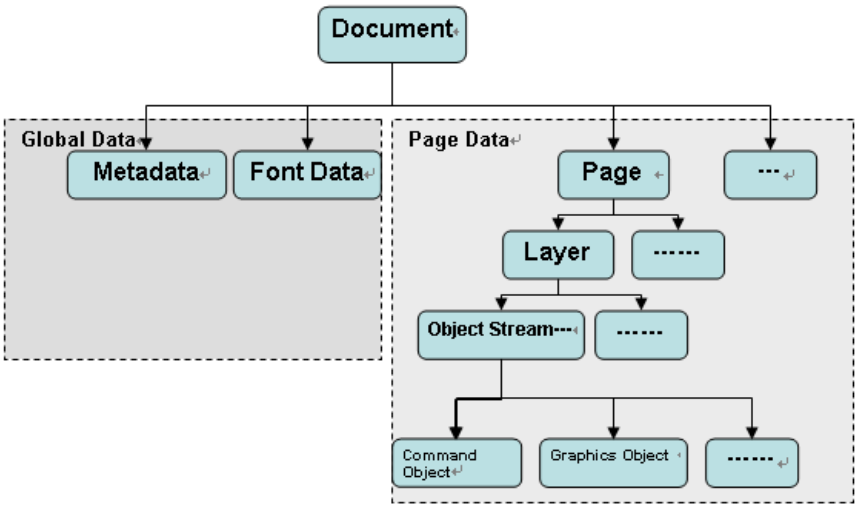


Figure 2. UOML Abstract Document Model 2

After

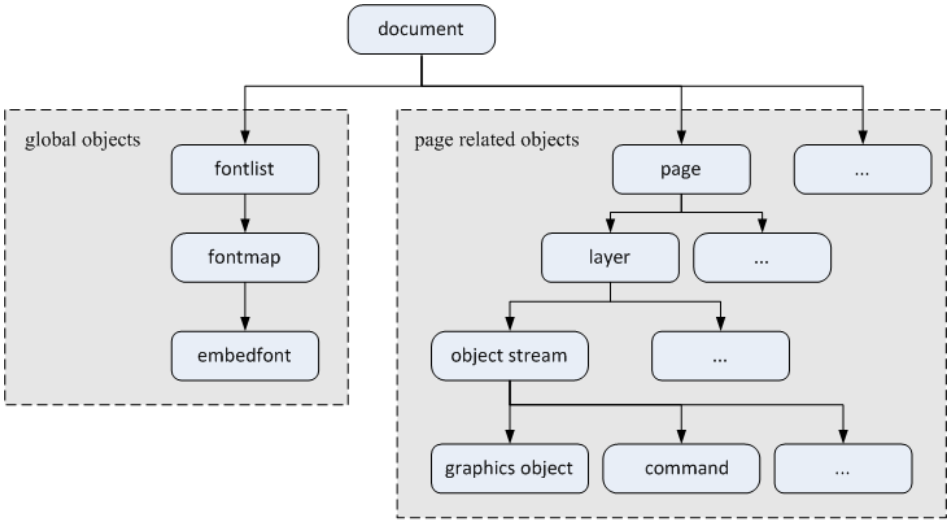


Figure 2. UOML Abstract Document Model 2

19. Move all of the UOML objects in Document Architecture (old) to the UOML Objects clause (new).

All specific UOML objects in the old Document Architecture clause (*docbase*, *docset*, *doc*, *metalist*, etc.) were moved to a new UOML Objects clause. These include current clauses 2.1.1, 2.1.2, 2.1.3, 2.3.1, 2.3.1.1, 2.3.1.2, 2.3.2.1, 2.3.2.2, 2.3.2.3, 2.4.1, 2.4.2, 2.4.3, 2.5.1 through 2.5.13, 2.15.13.1, 2.5.13.2, 2.6.1, 2.6.2, 2.6.2.1 through 2.6.2.35, 2.7. Propose to move these to a new clause 4.

20. Remove Internal Structure sub-clause.

Remove the Internal Structure of the Document sub-clause from the specification. Associated wording to be rewritten and repurposed to other clauses, specifically the Font and Page sub-clauses. (current clause 2.2; proposed new clauses 2.5, 2.6)

Before

Document is consisted of Document Global Data and Page Data.

Document Global Data can be used globalize among the document, it includes Metadata and Font. Metadata is the list for a set of keys and values, and is a sub-object of DOC, while Font is a list of font mappings (FONTMAP) and is also a sub-object of DOC. Each font mapping describes a font type used in the document, including font name, font sequential number defined in the document, and optional embedded Font.

Page Data may include 0 to multiple page(s) (PAGE). Each page has properties to describe width, height, resolution; each page may contain one to multiple layer(s) (LAYER), and each layer may contain one to multiple object stream; each object stream may contain a sequence of objects, which include graphics objects (details about this, see 2.5 Graphics objects) and non-displayable command objects（CMD）(details about this, see 2.6 Command Objects).

After

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.

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*]

21. Remove the phrase "Document Global Data" from the spec.

Remove the Document Global Data phrase from the spec. Metadata and Font remain as entities, but just not under the Document Global Data name. Document Global Data is repetitive and the discussion around Metadata and Font covers the discussion regarding Document Global Data. Most of the entries of Document Global Data have been removed due to associated changes above (e.g. errata #20). However it was also removed from current clause 2.1.3 and the entire current clause 2.3 has been removed.

22. Add a new Docbase sub-clause.

Add a description of the docbase object (not its direct implementation specification) to the proposed Abstract Document Model clause. (proposed new clause 2.2)

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.

Comment [JM35]: Accepted by GB-10

23. Add new Docset sub-clause.

Add description of the docset object (not its direct implementation specification) to the proposed Abstract Document Model clause (proposed new clause 2.3).

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.

Comment [JM36]: Accepted by GB-10

24. Add new Document sub-clause.

Add description of the Document object (not its direct implementation specification) to the proposed Abstract Document Model clause.(proposed new clause 2.4)

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*]

Comment [JM37]: DE-14, GB-26 accepted this.

Sub-object: fontlist, page.

25. Rename the Font Definition sub-clause to Font and edit it for better clarity.

.(current clause 2.3.2; proposed new clause 2.5)

Before

Font Definition

Font contains of a FONTLIST, the FONTLIST consists of 1 to multiple FONTMAP, and FONTMAP consists of zero or one EMBEDFONT. EMBEDFONT, FONTMAP, FONTLIST in the order given above, the previous one is the sub-object of the latter one,and can be generated by UOML's INSERT instruction.

After

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.

Comment [JM38]: GB-25 accepted this.

26. Rename the Page Data sub-clause to Page and edit for better clarity.

(current clause 2.4; proposed new clause 2.6)

Before

Page Data

Page data includes PAGE, LAYER, OBJSTREAM, Graphics Objects and Command Objects, please check Figure 2 for their structure.

LAYER is PAGE's sub-object, OBJSTREAM is LAYER's sub-object, Graphics Objects and Command Objects are OBJSTREAM's sub-object, they can be generated by UOML INSERT instruction. Graphics Objects refers all the visible objects, check 2.5 for details. Command Objects refers objects that can control graphics state of the render engine, check 2.6 for details.

After

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*]

Comment [JM39]: GB-26 accepted this.

Sub-object: layer.

27. Add a new Layer sub-clause.

A description of the Layer object (not its direct implementation specification) will be added to the proposed Abstract Document Model clause. (proposed new clause 2.7)

A layer object corresponds to one layer in a page. . 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*]

Comment [JM40]: As a result of the acceptance of GB-21 and GB-29

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

Sub-object: object stream.

28. Add a new Object Stream sub-clause.

Add a description of the Document object (not its direct implementation specification) to the proposed Abstract Document Model clause. (proposed new clause 2.8)

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.

Comment [JM41]: De-17 accepted proposed change.

Sub-object: graphics object, command object.

29. Rename Graphics Objects to Graphics Object sub-clause and edit for better clarity

(current clause 2.5; proposed new clause 2.9). A new Graphics Objects clause is being proposed to be added to the newly proposed clause 4 as clause 4.10. See errata #60 below.

Before

Graphics objects refer to the objects that could make render engine to draw, such as text, image, and Path. They describe the appearance of the page. The types and definitions of graphics objects are given as follows.

After

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.

30. Rename Command Objects to Command Object sub-clause and edit for better clarity.

(current clause 2.6; proposed new clause 2.10). An updated Command Objects clause is being proposed to be added to the newly proposed clause 4 as clause 4.11. See errata #61 below.

Before

Command objects are Page Data's sub-object, are used for modifying the current graphics state that holds current graphics control parameters, such as text size, typeface and color. The properties include command name, command value and other possible data (such as clip, transformation matrix, color and so on).

After

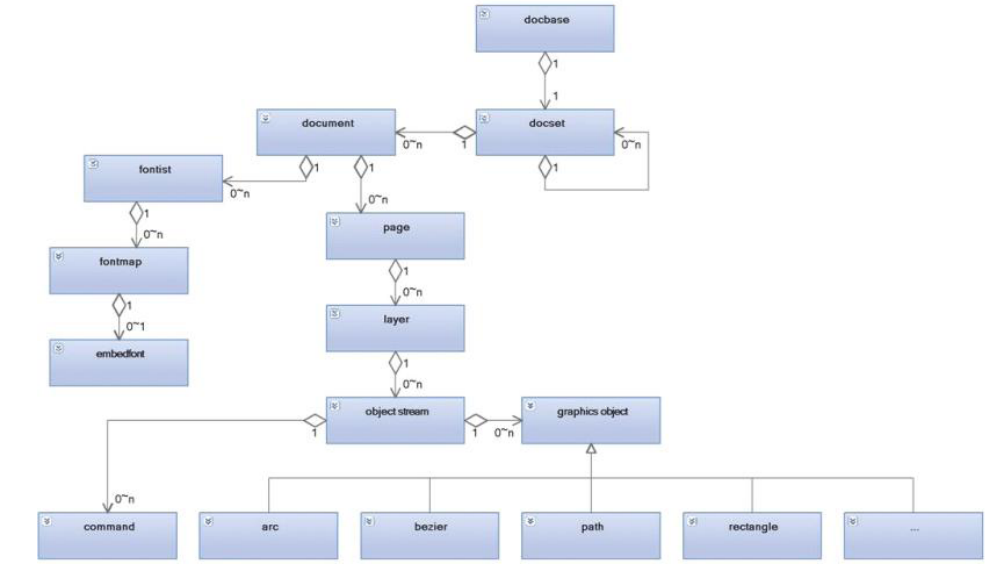
A command object changes one or multiple parameters in the current graphics state. The graphics state is initialized at the beginning of the rendering of each layer with the default values specified in section §4.13. The rendering of a graphics object relies on the current parameters in the graphics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

31. Add a UML Diagram of UOML sub-clause.

Comment [JM42]: Accepted by CZ-03 and DE-04

At the request of the NBs, a UML diagram is proposed for the abstract document model. (proposed new clause 2.11)

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.



32. Minor changes in the Page Rendering Model sub-clause.

(current clause 2.4.4; proposed new clause 2.12)

Before

Page rendering steps:

1. Repeat the following step from the first layer to the last layer.
    1. Initialize the current graphics state of the render engine with the default value (refer to 2.7).
    2. Loop through the object streams of the current layer.
        1. Then loop through the objects of each object stream.
            1. Draw the object if it is graphics object.
            2. Otherwise update the graphics state according to the content of the command object.
2. Page rendering completes.

After

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.

Comment [JM43]: GB-29 accepted this.

33. Move the Coordinate and Path Encoding Rules sub-clause to the proposed new UOML Objects clause.

Move the Coordinate and Path Encoding Rules sub-clause to the newly proposed UOML Objects clause. (current clause 2.5.12; proposed new clause 4.10.12)

34. Move the Definition of Referenced Type sub-to the proposed new UOML Objects clause.

Move the Definition of Referenced Type sub-clause to the newly proposed UOML Objects clause. (current clause 2.5.13; proposed new clause 4.11.3)

35. Move the Default Value of Graphics State sub-clause to the proposed new UOML Objects clause.

Move the Default Value of Graphics State sub-clause to the newly poposed UOML Objects clause. (current clause 2.7; proposed new clause 4.12)

36. Expand Introduction to the UOML Instructions clause and modify editorially.

(current clause 3; proposed new clause 3)

Before

UOML Instructions are used to define operations to UOML objects, such as create a Docbase, insert a sub-object, delete a object, change attribute of a object, etc.

This chapter will define the syntax and semantics of the following UOML instructions

1 OPEN
2 CLOSE
1 USE
2 GET
3 SET
4 INSERT
5 DELETE
6 SYSTEM
7 RET

After

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.

37. Modify all Example titles in the specification to a new format

The new format for examples would be:

[*Example*: …. *end example*]

Affects all clauses with examples. For example, in current clause 2.6.1, the change would be:

Before

EXAMPLE 1:

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

After

[*Example*:

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

end example]

38. Ensure all Examples in the document include namespace information in the outer elements

Here is an example of the change in the OPEN instruction, current clause 3.1:

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

39. Change all XML fragments to Courier Font.

40. Editorial Changes Were Made Throughout Many Instructions.

Minor editorial changes were made throughout all of the UOML instructions to bring them to better English grammatical compliance. (current clauses 3.1, 3.2, 3.6, 3.8; proposed new clauses 3.1, 3.2, 3.6, 3.8)

Before

OPEN

path: a character string value, representing the path of a Docbase.

CLOSE

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

INSERT

Semantics:
INSERT inserts an object under another one as its sub-object.

SYSTEM

Semantics:

SYSTEM executes system maintenance. Within this version, it has only one function: to save the docbase. It includes a sub-element:

flush—to save the Docbase

After

OPEN

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*]

CLOSE

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

INSERT

Semantics:

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

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*]

41. Add a Note to the Semantics of the USE Instruction.

(current clause 3.3; proposed new clause 3.3)

[*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*]

42. Reformat the GET Instruction.

(current clause 3.4; proposed new clause 3.4)

Before

Semantics:

GET can retrieve the sub-object handle, the count of sub-objects, the property value of an object, a page bitmap.

Properties:

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

handle: a character string value, representing object handle of the current operation. It is optional, if not exists, use the handle set by USE instead.

Sub-elements:

pos: used when usage=GET_SUB. It has one property listed below.

Property of this sub-element:

val: represents position number of the specified sub-object, starts from 0.

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

property: used when usage=GET_PROP. It has one property listed below.

Property of this sub-element:

name: represents the getting property name.

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

disp_conf: used when usage=GET_PAGE_BMP. It has five properties and one option sub-element listed below.

Properties of this sub-element:

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

resolution: represents resolution of bitmap

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

output: represents whether to put out to the file or to the memory and the value to be chosen is 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:

return value based on usage value.

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

GET_SUB: If the usage is GET_SUB, indicate to get the handle of some specific sub-object. In this case, GET contains sub-element of 'pos'. The return value, which is within RET, contains one 'stringVal' sub-element, and its 'name' property is handle, its 'val' property presents this sub-object's handle

GET_PROP: If the usage is GET_PROP, indicate to get some specific property of this specific object. In this case UOML instruction GET shall contain sub-element of 'property'; If the operation succeeds, the sub-element of return value, which is within RET, is not certain and the concrete sub-element name relies on the type it has got, the 'name' property of the sub-element is the property name to get, 'val' property is the value of the property.

GET_PAGE_BMP: If the usage is GET_PAGE_BMP,indicate to get the this specific page bitmap In this case, GET shall contain sub-element 'disp_conf'; It only use return value defined by RET When it fails, the return value is defined by RET.

EXAMPLE 1: get the total number of sub-objects of the specific object

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

EXAMPLE 2: get a specific sub-object handle

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

EXAMPLE 3: get specific property of the object

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

EXAMPLE 4: 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>
```

After

Semantics:

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

Properties:

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

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

Sub-elements:

pos: used when usage=GET_SUB.

Property of this sub-element:

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

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

property: used when usage=GET_PROP.

Property of this sub-element:

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

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

disp_conf: used when usage=GET_PAGE_BMP.

Properties of this sub-element:

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

resolution: represents resolution of bitmap

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

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

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

Sub-element of this sub-element:

clip: represents clip area for output, PATH type.

Usage value / Return value:

The return value is based on the usage value:

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

[*Example*:

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

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

RET instruction returns the number:

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

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

RET instruction returns the handle of the sub-object:

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

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

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

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

```
<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>
</GET>
```

end example]

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

[*Example*:

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

end example]

43. Modify the pos: property in the INSERT instruction to clarify position numbers

(current clause 3.6; proposed new clause 3.6)

Before

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.

After

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.

Comment [JM44]: Accepted by comment GB-29

44. Modify the semantics of DELETE instruction to clarify position numbers

(current clause 3.7; proposed new clause 3.7)

Before

DELETE deletes an object.

After

pos: DELETE deletes and 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*]

Comment [JM45]: Accepted by comment GB-29

45. Add a note to USE describing object and current object

(current clause 3.7; proposed new clause 3.7)

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*]

Comment [JM46]: KR-03 accepted this.

46. Clearly specify where the normative UOML schema is located.

This will be placed in the normative references clause as well as the newly proposed UOML Objects clause (proposed new clause 4)

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

47. Add a new UOML Objects clause with introductory wording.

(Proposed new clause 4)

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

48. Add a New Logical Coordinate System and Units sub-clause.

(Proposed new clause 4.1)

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

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*]

Comment [JM47]: CZ-05 accepted proposal

Comment [JM48]: De-16 accepted this.

49. Add a New Graphics State sub-clause.

(Proposed new clause 4.2)

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.

Comment [JM49]: GB-30 accepted this

50. Editorial changes to some of the UOML object semantics, properties, etc.

Editorial changes were made throughout the entire UOML objects clause to bring them to better English grammatical compliance, as well as to enhance clarity. Referenced clauses in the edited text refer to proposed clauses. (current clauses 2.1.1, 2.1.2, 2.1.3, 2.4.1, 2.4.2, 2.4.3, 2.5.1; proposed new clauses 4.3, 4.4, 4.5, 4.7, 4.8, 4.9, 4.10.1)

Before

DOCBASE

Semantics: UOML document structure's top level. It has only one root DOCSET, other DOCSET and DOC are the root DOCSET's direct or indirect sub-objects. The root DOCSET will be generated automatically when the DOCBASE is created.

Properties:

name: name of Docbase.

path: location of the Docbase.

DOCSET

Semantics: a set of DOC and/or DOCSET.

DOC

Semantics: refers layout-based document, a single document has 0 to multiple pages

PAGE

Semantics: page within document.

LAYER

Semantics: layers inside page.

OBSTREAM

Semantics: a sequence of objects.

ARC

Semantics:

ARC includes five properties: starting point, ending point, center, direction and angle.

Properties:

start: starting position of arc.

end: ending position of arc.

center: center of ellipse arc.

clockwise: direction for arc is from the starting point to the ending point, which can be clockwise or anti-clockwise. It is a Boolean value, "true" means clockwise, "false" means anti-clockwise.

angle: inclination from coordinate system's x-axis to arc's x-axis. It is calculated by radian. Positive value means anti-clockwise, negative value means clockwise.

After

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.

DOCSET

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

DOC

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

PAGE

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

LAYER

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

OBSTREAM

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

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.

51. Remove all Sub-Object items from the UOML object descriptions.

See errata #19 regarding all of the current clauses that are affected. SomeSub-object items are in the sub-clauses of the newly proposed Abstract Document Model clause (proposed clause 2.x).

52. Change the semantics of objects to reflect the descriptions provided in the newly proposed Abstract Document Model clause.

(current clauses: 2.1.1, 2.1.2, 2.1.3, 2.3.1.1, 2.4.1, Proposed new clauses 4.3, 4.4, 4.5, 4.5.1, 4.7)

Before

DOCBASE

Semantics: UOML document structure's top level. It has only one root DOCSET, other DOCSET and DOC are the root DOCSET's direct or indirect sub-objects. The root DOCSET will be generated automatically when the DOCBASE is created.

DOCSET

Semantics: a set of DOC and/or DOCSET.

DOC

Semantics: refers layout-based document, a single document has 0 to multiple pages.

Metadata

General information, such as the document's title, author, creation and modification date, is called metadata. Metadata consists of a METALIST, the METALIST consists of 1 to multiple META.

PAGE

Semantics: page within document.

After

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

DOCSET

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

Properties:

name: name of docset.

Sub-elements: N/A

DOC

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

Properties:

name: name of document.

Sub-elements:

metainfo: metadata of the document, METALIST type.

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.

Comment [JM50]: GB-25, GB-27 accepted this.

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).

Comment [JM51]: DE-16 accepted this.

Sub-elements: N/A

53. Move Font Definition from old Document Architecture clause to proposed UOML Object clause and re-edit.

(current clauses: 2.3.2, 2.3.2.1, 2.3.2.2, 2.3.2.3, Proposed new clauses 4.6, 4.6.1, 4.6.2, 4.6.3)

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

Comment [JM52]: GB-28 accepted this.

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

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

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

54. Move the originally specified “Note:” Items to the semantics.

In the original specification, “Note:” in object definitions were placed after the properties, sub-elements and/or sub-objects. However, these will now be moved as informative notes within the actual semantic wording. Thus anywhere, you see a “Note:”, it will be moved to the semantics. For example, here is the change to IMAGE, current clause 2.5.5:

Before

Semantics:

Image includes four properties: coordinates of top-left corner, coordinates of bottom-right corner, image type, image file pathname or inline image data.

Properties:

tl: coordinates of top-left corner as the image is displayed in the page

br: coordinates of the bottom-right corner as the image is displayed in the page

type: image type, possible value includes "bmp", "png", "jpeg", "jbig", "tiff", representing BMP, PNG, JPEG, JBIG, TIFF image respectively.

*path(*optional*):* path of the image file. If present, the content of IMAGE element should leave blank, otherwise the content of IMAGE element contains the base64 encoded raw image data.

Sub-element: N/A

Sub-object: N/A

Note:

Image may contains large amount of bytes, and it will greatly reduce the performance of XML parser, it is recommended to transfer large image using a file by specifying the filename in the 'imgpath' property.

After

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 (e.g, path). *end note*]

Comment [JM53]: GB-34 accepted this.

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

55. Remove the Page Data sub-clause, current clause 2.4

56. Add Units for PAGE properties.

(current clause: 2.4.1, Proposed new clause 4.7)

Before

Properties:

width: width of the page.

height: height of the page.

resolution: resolution of the page.

After

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).

Comment [JM54]: FR-03 rejected this, but comments such as DE-16 and DE-26 accepted it. So accepting this change.

57. Remove a sentence from the Graphics Objects sub-clause.

(current clause: 2.5, Proposed new clause 4.10)

Before

Graphics objects refer to the objects that could make render engine to draw, such as text, image, and Path. They describe the appearance of the page. The types and definitions of graphics objects are given as follows.

After

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

58. Reword the Command Objects sub-clause.

(current clause: 2.6, Proposed new clause 4.11)

Before

Command objects are Page Data's sub-object, are used for modifying the current graphics state that holds current graphics control parameters, such as text size, typeface and color. The properties include command name, command value and other possible data (such as clip, transformation matrix, color and so on).

After

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

59. Change the description of the angle property of ARC

(current clause: 2.5.1, Proposed new clause 4.10.1)

Before

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

After

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.

Comment [JM55]: GB-33 accepted this.

60. Change the semantics wording of BEIZER

(current clause: 2.5.2, Proposed new clause 4.10.2)

Before

Bezier curve includes four properties: starting point, control point 1, control point 2 and ending point.

After

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.

61. Add valid ranges and units to the properties of various objects

(current clauses: 2.5.3, 2.5.4, 2.5.8, 2.6.2.6, 2.6.2.9, 2.6.2.16, 2.6.2.17, 2.6.2.26, 2.6.2.27, 2.6.2.33, 2.6.2.34, 2.6.2.35. Proposed new clauses 4.10.3, 4.10.4, 4.10.8, 4.11.2.6, 4.11.2.9, 4.11.2.16, 4.11.2.17, 4.11.2.26, 4.11.2.27, 4.11.2.33, 4.11.2.34, 4.11.2.35)

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

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.

Comment [JM56]: CZ-05 accepted proposed change.

Sub-elements: N/A

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

Comment [JM57]: GB-35 accepted this.

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

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

CHAR_SLANT

Semantics: Set the slant of the current character.

Properties:

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

Sub-elements: N/A

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

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

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

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

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

HOLLOW_BORDER

Semantics: Set the border width of the current hollow character

Properties:

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

Comment [JM58]: FR-03 rejected this, but CZ-05 accepted it. Thus going with acceptance.

62. Modify wording in PATH to clarify that it is the actual element and not conceptual.

(current clause: 2.5.10, Proposed new clause 4.10.10)

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.

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

end example].

Comment [JM59]: Accepted by comment GB-12.

Comment [JM60]: Accepted by comment GB-37

63. Add an example in SUBPATH for clarity

(current clause: 2.5.9, Proposed new clause 4.10.9)

[*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.

```
<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>
</INSERT>
```

end example].

Comment [JM61]: Accepted by GB-36

64. Add further description for the properties of TEXT.

(current clause: 2.5.11, Proposed new clause 4.10.11)

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.

Comment [JM62]: GB-38 accepted this.

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.

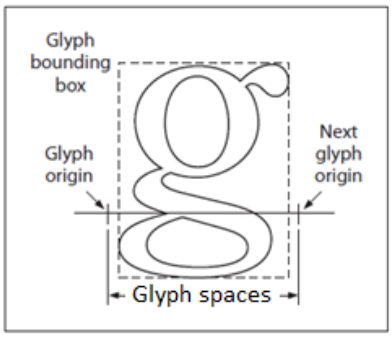


Figure 4. spaces of text

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

Comment [JM63]: DE-21 accepted this

65. Add thickness to the semantics of LINE_WIDTH.

(current clause: 2.6.2.6, Proposed new clause 4.11.2.6)

Semantics: set the current line width/thickness

66. Clarify wording in the RASTER_OP properties

(current clause: 2.6.2.12, Proposed new clause 4.11.2.12)

Before

Here, pixel color is the color after rendering; 'src' is the currently used color, also called the foreground color; 'dest' is the current color of the page, also called background. '&' is bitwise AND,'|' is bitwise OR,'^' is bitwise XOR,'~' is bitwise NOT, which has the highest priority.

After

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.

67. Restructure the semantics of TEXT_DIR to show text direction is from beginning to end.

(current clause: 2.6.2.13, Proposed new clause 4.11.2.13)

Before

Semantics: set the current text direction. The direction is from the end of a text row to the beginning.

After

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.

68. Clarify the meaning of character direction in the CHAR_DIR clause

(current clause: 2.6.2.14, Proposed new clause 4.11.2.14)

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

Comment [JM64]: GB-46 accepted this.

69. Add wording saying the v1 property in CHAR_SLANT is regardless of reading direction.

(current clause: 2.6.2.16, Proposed new clause 4.11.2.16)

Properties:

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

Comment [JM65]: GB-47 accepted this.

70. Provide default value information and cardinalities for CHAR_WEIGHT.

(current clause: 2.6.2.18, Proposed new clause 4.11.2.18)

Before

Semantics: set the current character weight.

Properties:

v1: a floating point number, from 0 to 1, representing the character weight.

Sub-elements: N/A

After

Semantics: Set the current character weight. The default value is 0. The thickness of a character stoke 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

71. Add further details for the possible values of the v1 property of CHAR_STYLE.

(current clause: 2.6.2.19, Proposed new clause 4.11.2.19)

CHAR_STYLE

Semantics: Set the current character style.

Properties:

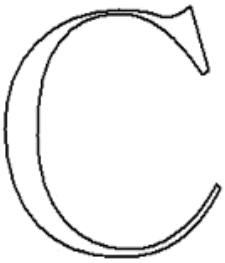
v1: a character string, representing the character style. The possible values for this property are SHADOW, HOLLOW and OUTLINE, or some combination of the three, separated by commas. If the string is set to empty, then any previous setting is cleared.

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

Comment [JM66]: DE-25 accepted this. GB-49 accepted this too.

72. Add that the TEXT_MATRIX object applies to each character individually.

(current clause: 2.6.2.20, Proposed new clause 4.11.2.20)

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:

Comment [JM67]: GB-50 accepts this.

73. Add that SHADOW_WIDTH represents the thickness of the outline of a shadow.

(current clause: 2.6.2.26, Proposed new clause 4.11.2.26)

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

74. Add that SHADOW_LEN represents the displacement of the shadow with respect to the character.

(current clause: 2.6.2.27, Proposed new clause 4.11.2.27)

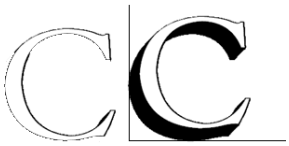
Semantics: Set the length of the current character shadow. SHADOW_LEN represents the displacement of the shadow with respect to the character.

Comment [JM68]: DE-25 accepted this.

75. Added an illustrative example to SHADOW_ATL.

(current clause: 2.6.2.29, Proposed new clause 4.11.2.29)

[*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]

Comment [JM69]: GB-52 accepted this.

76. Add captions to the SHADOW_NEG picture illustrations.

(current clause: 2.6.2.30, Proposed new clause 4.11.2.30)

SHADOW_NEG is false

SHADOW_NEG is true

Comment [JM70]: GB-52 accepted this.

77. Add further explanation to the CLIP_AREA specification.

(current clause: 2.6.2.31, Proposed new clause 4.11.2.31)

Before

Semantics: set the current clip area

Properties: N/A

Sub-elements:

cliparea: PATH type, representing the clip area

After

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.

Comment [JM71]: DE-25 accepted this. GB-53 accepted this.

78. Create a table for the Default Value of Graphics State.

(current clause: 2.7, Proposed new clause 2.7)

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

79. Add a short introduction to the Definition of Referenced Type sub-clause.

(current clause: 3.10, Proposed new clause 2.15.13)

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

80. Add a Data Ranges sub-clause.

Provided general rules for data ranges and cardinalities of objects. It is important to remember that the normative XML schema provides information on data and data types. (Proposed new clause 4.14)

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 integer 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 integer values between 0-255.
7. Valid ranges and formats for a date are as defined by xs:date in XML Schema 1.0 Part 2.

81. Expand the Conformance clause.

(current clause: 5, Proposed new clause 5)

Before

In order to conform to this specification, an implementation:

- SHALL support all the functional and interface requirements defined in this specification.
- SHALL NOT specify any requirements that would contradict or cause non-conformance to this specification.

A conformance implementation SHALL satisfy the conformance requirements of the applicable parts of this specification.

After

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).

DCMS Conformance

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

Comment [JM72]: FR-03, KR-04 rejected this. However, CZ-05 accepted this addition with proposed changes. So we are accepting this clause. DE-25 accepted this as well.

Comment [JM73]: Changes as specified by CZ-05

Comment [JM74]: GB-58 accepted this.

this OASIS standard.

Application Conformance

A 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.

Comment [JM75]: CZ-01 accepted proposed changes.

Comment [JM76]: FR-05 rejected this, but DE-30 accepted it. NL-03 also accepted this. Going with the accepted.

82. Add an example to the PATH object

(current clause: 2.5.10, Proposed new clause: 4.10.10)

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

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

end example]

83. Expand and add to all inline specification examples to be more informational.

The examples that are inline to the specification (see errata below for a new proposed stand-alone Annex containing more examples) should be modified to be more informational and correct. (current clause: 2.6.1, 3.1 through 3.9, Proposed new clauses: 4.11.1, 3.1 through 3.9)

Before

CMD

EXAMPLE 1:

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

```
</CMD>
```

EXAMPLE 2:

```
<CMD name="LINE_CAP" v1="END_BUT"/>
```

EXAMPLE 3:

```
<CMD name="TEXT_MATRIX">
    <matrix f11="2" f12="0" f21="0" f22="1.5" f31="10" f32="20"/>
</CMD>
```

OPEN

EXAMPLE: create a docbase, named 1.sep

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

CLOSE

EXAMPLE: close a docbase.

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

USE

EXAMPLE: set up the handle represented object as current object.

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

GET

EXAMPLE 1: get the total number of sub-objects of the specific object

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

EXAMPLE 2: get a specific sub-object handle

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

EXAMPLE 3: get specific property of the object

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

EXAMPLE 4: 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>
```

SET

EXAMPLE: set specific object's angle property.

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

INSERT

EXAMPLE 1: insert a text data

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

EXAMPLE 2:insert a layer.

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

DELETE

EXAMPLE: delete an object

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

SYSTEM

EXAMPLE: save the Docbase example.sep

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

RET

EXAMPLE: Return three values.

```
<uoml:RET>
    <intVal name="xxx" val="0"/>
    <boolVal name="SUCCESS" val="false"/>
    <stringVal name="ERR_INFO" val="required resource not available"/>
</umol:RET>
```

After

CMD

[*Example*:

```
<INSERT pos=”2” handle=”vs03”>
```

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

end example]

[*Example*:

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

end example]

[*Example*:

```
  <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>
  </INSERT>
```

end example]

OPEN

[*Example*:

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

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

      Return element if OPEN succeeds:

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

      Return element if OPEN fails:
```

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

end example]

CLOSE

[*Example*:

Close a docbase.

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

end example]

USE

[*Example*:

Set up the handle represented object as the current object.

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

end example]

GET

[*Example*:

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

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

RET instruction returns the number:

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

end example]

[*Example*:

Get a specific sub-object handle:

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

RET instruction returns the handle of the sub-object:

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

end example]

[*Example*:

Get specific property of the object

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

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

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

end example]

[*Example*:

Get specific page's bitmap

```
<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>
</GET>
```

end example]

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

[*Example*:

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

end example]

SET

[*Example*:

Set specific object's angle property.

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

end example]

INSERT

[*Example*:

Insert text data

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

end example]

[*Example*:

Insert a layer

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

end example]

DELETE

[*Example*:

Delete an object

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

end example]

SYSTEM

[*Example*:

Save the docbase example.sep

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

end example]

RET

[*Example*:

Return two values.

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

end example]

84. Acknowledgments move from Annex A to Annex C.

With the following additions: (current clause: Annex A, Proposed new clause: Annex C)

Ningsheng Liu (Sursen Corporation)

Kaihong Zou (Sursen Corporation)
Joel Marcey (Sursen Corporation)

85. Add a copy of the UOML XML Schema as Annex A.

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"/>
```

Comment [JM77]: Changed as a result of the acceptance of GB-08. This informative schema will be made normative.

Comment [JM78]: Changed as a result of the acceptance of GB-08. This informative schema will be made normative.

```
            <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>
```

Comment [JM79]: KR-01 rejected this, but GB-08 accepted this with proposed changes. And GB-15 accepted this.

End of informative text.

86. Add Detailed UOML Examples as Annex B.

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.

Comment [JM80]: This new appendix is accepted as a result of GB-09. Also NL-03 accepted this.

Also, the outer element namespace additions were added as a result of the acceptanc of GB-13 , DE-05 and CZ-06

87. Add a RELAX NG Representation of the UOML XML Schema as an Annex C.

Comment [JM81]: This is added as the result of the partially accepted change for JP-10

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 }?,
   ((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.