UOML (Unstructured Operation Markup Language) Part 1 Version 1.0

OASIS Standard

10 October 2008

Specification URIs:
This Version:

http://docs.oasis-open.org/uoml-x/v1.0/os01/uoml-part1-v1.0-os01.pdf (Authoritative)

http://docs.oasis-open.org/uoml-x/v1.0/os01/uoml-part1-v1.0-os01.odt

http://docs.oasis-open.org/uoml-x/v1.0/os01/uoml-part1-v1.0-os01.html

Previous Version:

http://docs.oasis-open.org/uoml-x/v1.0/cs01/uoml-part1-v1.0-cs01.pdf (Authoritative)

http://docs.oasis-open.org/uoml-x/v1.0/cs01/uoml-part1-v1.0-cs01.odt

http://docs.oasis-open.org/uoml-x/v1.0/cs01/uoml-part1-v1.0-cs01.html

Latest Version:

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

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

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

Technical Committee:

OASIS Unstructured Operation Markup Language Extended (UOML-X) Technical Committee

Chair(s):

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

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

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

Editor(s):

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

Yan Allison Shi, Sursen Corp. <allison_shi@sursen.com>

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

Related work:

[N/A]

Declared XML Namespace(s):

urn:oasis:names:tc:uoml:xmlns:uoml-x:1.0 (prefix: uoml)

Abstract:

This document defines a markup language for unstructured document operation, including the

definitions of abstract document model and document operating instructions to the abstract document model.

Status:

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

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

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

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

Notices

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

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

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

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

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

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

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

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

Table of Contents

1. Introduction

1.1. Terminology

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:

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.

1.2. 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.

1.3. 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

1.4. Non-Normative References

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

2. UOML Document Structure

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

2.1. Document Architecture

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.

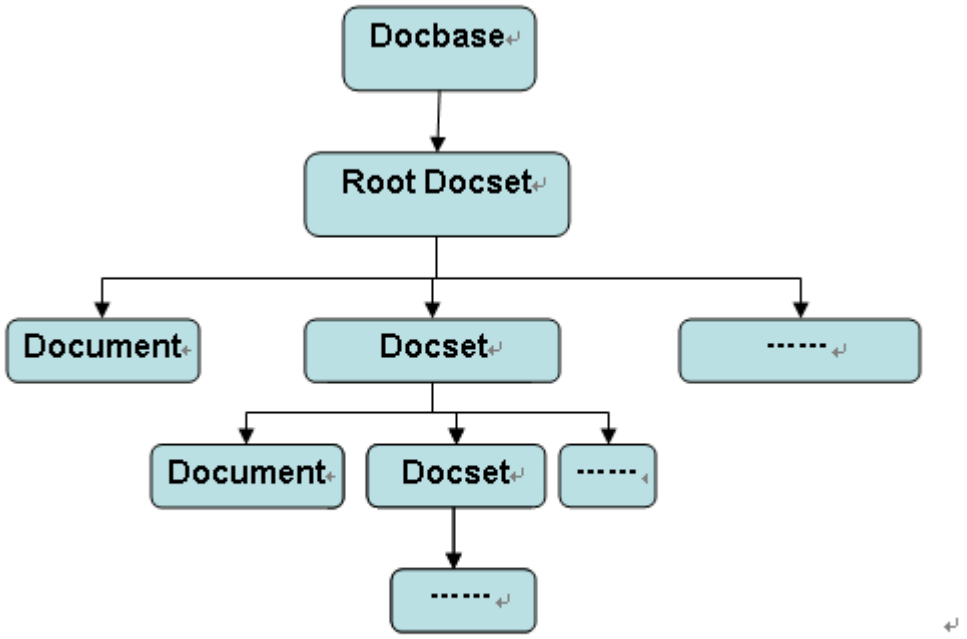


Figure 1. UOML Abstract Document Model 1

2.1.1. 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.

Sub-element: N/A

Sub-object: root DOCSET.

2.1.2. DOCSET

Semantics: a set of DOC and/or DOCSET.

Properties:

name: name of Docset.

Sub-element: N/A

Sub-object: DOC, DOCSET.

2.1.3. DOC

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

Properties:

name: name of document.

Sub-element:

metainfo: metadata of the document, METALIST type.

Sub-object: Page Data and Document Global Data.

2.2. Internal Structure of Document

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).

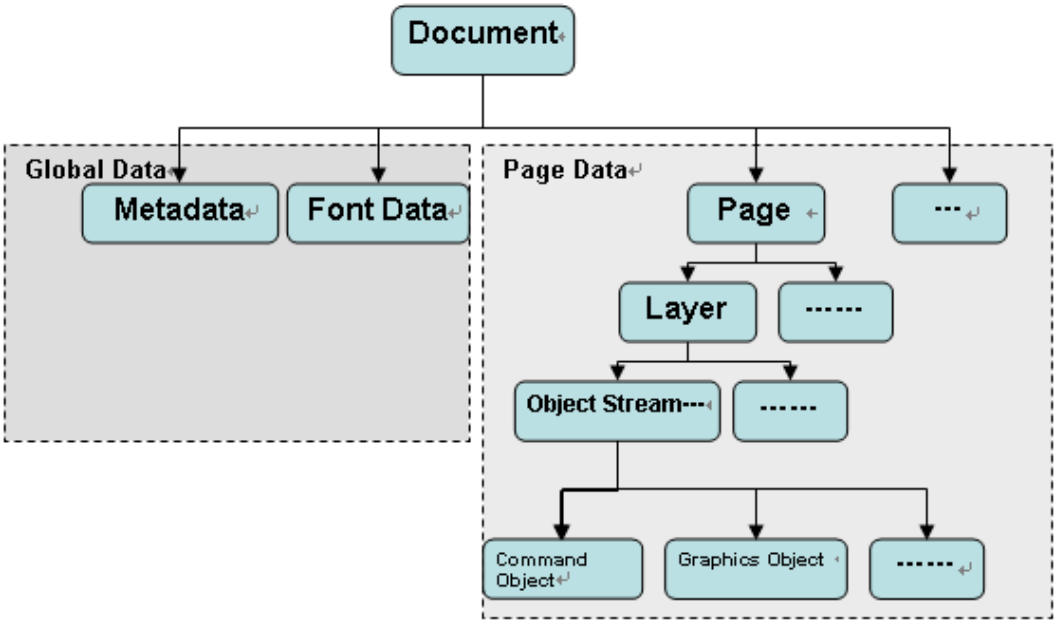


Figure 2. UOML Abstract Document Model 2

2.3. Document Global Data

Document Global Data includes Metadata and Font definition.

2.3.1. 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.

2.3.1.1.METALIST

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

Properties: N/A

Sub-element:

meta: META type.

Sub-object: N/A

2.3.1.2. META

Semantics: one metadata, including two properties: key and value.

Properties:

key: character string value which presents the key of metadata.

val: character string value which presents the value of metadata.

Sub-element: N/A

Sub-object: N/A

2.3.2. 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.

2.3.2.1. FONTLIST

Semantics: a list of all the fonts used in the document

Properties: N/A

Sub-element: N/A

Sub-object: FONTMAP

2.3.2.2. FONTMAP

Semantics: defines one font used in the document.

Properties:

name: name of the font

no: the id of the font quoted in document.

Sub-element: N/A

Sub-object: EMBEDFONT

2.3.2.3. EMBEDFONT

Semantics: one embedded font type.

Properties: N/A

Sub-element: N/A

Sub-object: N/A

Note:

Use OpenFont as embedded font type. After encoding OpenFont using base64 format, put the result into EMBEDFONT's content section as the embed font data

2.4. 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.

2.4.1. PAGE

Semantics: page within document.

Properties:

width: width of the page.

height: height of the page.

resolution: resolution of the page.

Sub-element: N/A

Sub-object: 1 to multiple LAYER

2.4.2. LAYER

Semantics: layers inside page.

Properties: N/A

Sub-element: N/A

Sub-object: 0 to multiple OBJSTREAM.

2.4.3. OBJSTREAM

Semantics: a sequence of objects.

Properties: N/A

Sub-element: N/A

Sub-object: 0 to multiple Graphic Objects (see details on 2.5) and / or Command Objects (see details on 2.6).

2.4.4. Page Rendering Models

Page rendering steps:

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

2.5. Graphics Objects

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.

2.5.1. 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.

Sub-element: N/A

Sub-object: N/A

2.5.2. BEZIER

Semantics:

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

Properties:

start: starting point of Bezier curve.

ctrl: the first control point of Bezier curve.

ctrl2: the second control point of Bezier curve, it is optional.

end: ending point of Bezier curve.

Sub-element: N/A

Sub-object: N/A

Note:

The above is the definition of a third-order Bezier curve. However, if ctrl2 doesn't exist, the curve then becomes a second-order Bezier curve.

2.5.3. CIRCLE

Semantics:

Circle includes two properties: center and radius.

Properties:

center : coordinate of circle center

radius: radius of circle

Sub-element: N/A

Sub-object: N/A

2.5.4. ELLIPSE

Semantics:

Ellipse includes four properties: center, radius x, radius y and rotation angle.

Properties:

center: coordinates of ellipse center.

xr: length of radius x.

yr: length of radius y.

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

Sub-element: N/A

Sub-object: N/A

2.5.5. IMAGE

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.

2.5.6. LINE

Semantics:

Line has two properties: starting point and ending point

Properties:

start: coordinates of where the line starts

end: coordinates of where the line ends

Sub-element: N/A

Sub-object: N/A

2.5.7. RECT

Semantics:

Rectangle has two properties: coordinates of the top-left corner and coordinates of bottom-right corner.

Properties:

tl: coordinates of the top-left corner as the rectangle is shown in the page

br: coordinates of the bottom-right corner as the rectangle is shown in the page

Sub-element: N/A

Sub-object: N/A

2.5.8. ROUNDRECT

Semantics:

Round rectangle has four properties: coordinates of the top-left corner, coordinates of the bottom-right corner, radius x and radius y.

Properties:

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

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

xr: radius x of the round corner

yr: radius y of the round corner

Sub-element: N/A

Sub-object: N/A

2.5.9. SUBPATH

Semantics:

Sub-path refers to an irregular chain of curves consists of lines, Bezier curves and arcs, it can be either closed or not.

Properties:

data: defines the starting point and ending point of the sub-path, as well as each curve segment which makes up the sub-path.

Sub-element: N/A

Sub-object: N/A

Note: for encoding of property 'data', please refer to 2.5.12

2.5.10. PATH

Semantics:

Path refers to the open or closed region collection consist of one or multiple sub-path(s), circle(s), ellipse(s), rectangle(s) and round rectangle(s). PATH itself does not have data.

Properties: N/A

Sub-element:

circle: CIRCLE type, defines a circle.

ellipse: ELLIPSE type, defines ellipse.

rect: RECT type, defines rectangle.

roundrect: ROUNDRECT type, defines rectangle with round corners.

subpath: SUBPATH type, defines sub-path.

Sub-object: N/A

2.5.11. TEXT

Semantics:

Text has four properties: origin, encoding information, text data and character spacing list.

Properties:

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

encode: character set or encoding of text data.

text: character data contained in text, string data.

spaces: defines distances between adjacent characters' origin, separated by comma.

Sub-element: N/A

Sub-object: N/A

2.5.12. The Coordinate and Path Encoding Rules

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

Coordinates encoding rules

<coord> ::= <coord_x>,<coord_y>
<coord_x> ::= <int>
<coord_y> ::= <int>

In this Backus-Naur Form rule expression, 'coord' is coordinates, 'coord_x' is coordinate x, and 'coord_y' is coordinate y; <int> represents a string form of integer number.

Path encoding rules

<path> ::= <start> { <blank> (<line> | <bezier2> | <bezier3> | <arc>) }
<start> ::= s <blank> <coord>
<line> ::= l <blank> <coord>
<bezier2> ::= b <blank> <coord> <blank> <coord>
<bezier3> ::= B <blank> <coord> <blank> <coord> <blank> <coord>
<arc> ::= a <clockwise> <blank> <angle> <blank> <coord> <blank> <coord>
<clockwise> ::= true | false

<angle> ::= <float>

Semantics Definition

'coord' represents coordinates, refer to its previous definition.

'start' represents start point of sub-path.

'line' represents line segment.

'bezier2' represents second-order Bezier curve.

'bezier3' represents third-order Bezier curve.

'<blank>' represents one or multiple blank(s) or equivalent character, such as tab.

In the definition of 'line', 'coord' represent the ending point.

In the definition of 'bezier2', two 'coord' are for the control point and the ending point.

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

In the definition of 'arc', two 'coord' are center and end point.

Note:

Start point of each curve is the previous end point.

2.5.13. Definition of Referenced Type

2.5.13.1.COLOR_RGB

Semantics: color setting

Properties:

r: red component

g: green component

b: blue component

a: alpha component, it is optional.

Sub-element: N/A

Sub-object: N/A

2.5.13.2. MATRIX

Semantics: transformation matrix

Properties:

f11: floating point number

f12: floating point number

f21: floating point number

f22: floating point number

f31: floating point number

f32: floating point number

Sub-element: N/A

Sub-object: N/A

Note:

A transformation of matrix in UOML shall be specified by six numbers. In its most general form, this

array is denoted [f11 *f12 f21 f22 f31 f32*]; it can represent any linear transformation from one coordinate system to another. The transformation is carried out as follows:

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

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

2.6. Command Objects.

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).

2.6.1. CMD

Semantics: non-displayable command objects

Properties:

name: name of the command, refer to 2.6.2 to 2.6.37 for its possible value..

v1: optional command value.

v2: optional command value.

Sub-element:

rgb: COLOR_RGB type, used when 'name' is one of COLOR_LINE, COLOR_FILL, COLOR_SHADOW, COLOR_OUTLINE or COLOR_TEXT, refer corresponding sections below.

matrix: MATRIX type, used when 'name' is one of TEXT_MATRIX, IMAGE_MATRIX, GRAPH_MATRIX or EXT_MATRIX, refer corresponding sections below.

cliparea: PATH type, used when 'name' is CLIP_AREA, refer corresponding section below.

Sub-object: N/A

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>
```

2.6.2. CMD's name property values

Following sections describes what values can be used for 'name' property, and properties and sub-elements can be used for each valid 'name' value.

For example, if the CMD's 'name' property is 'COLOR_LINE', then CMD's sub-element is 'rgb'.

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

2.6.2.1. COLOR_LINE

Semantics: set the current line color

Properties: N/A

Sub-elements:

rgb: element of COLOR_RGB type, rgb specifies the color used to stroke lines and curves.

2.6.2.2. COLOR_FILL

Semantics: set the current fill color

Properties: N/A

Sub-elements:

rgb: element of COLOR_RGB type, rgb specifies the color used to fill an area.

2.6.2.3. COLOR_SHADOW

Semantics: set the current character shadow color

Properties: N/A

Sub-elements:

rgb: element of COLOR_RGB type, rgb specifies the color used to draw shadow of characters.

2.6.2.4. COLOR_OUTLINE

Semantics: set the current character outline color

Properties: N/A

Sub-elements:

rgb: element of COLOR_RGB type, rgb specifies the color used to draw outline of characters.

2.6.2.5. COLOR_TEXT

Semantics: set the current text color

Properties: N/A

Sub-elements:

rgb: element of COLOR_RGB type, rgb specifies the color used to draw characters.

2.6.2.6. LINE_WIDTH

Semantics: set the current line width

Properties:

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

Sub-elements: N/A

2.6.2.7. LINE_CAP

Semantics: set the current line cap style

Properties:

v1: a character string, representing line cap style. Possible value includes END_BUT, END_ROUND and END_SQUARE.

Sub-elements: N/A

Note:

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

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

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

2.6.2.8. LINE_JOIN

Semantics: set the current line join style

Properties:

v1: a character string, representing line join style. Possible value includes JOIN_MITER, JOIN_BEVEL and JOIN_ROUND

Sub-elements: N/A

Note:

JOIN_MITER: the outer edges of the strokes for the two segments shall be extended until they meet at too sharp an angle, a JOIN_BEVEL shall be used instead.

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

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

2.6.2.9. MITER_LIMIT

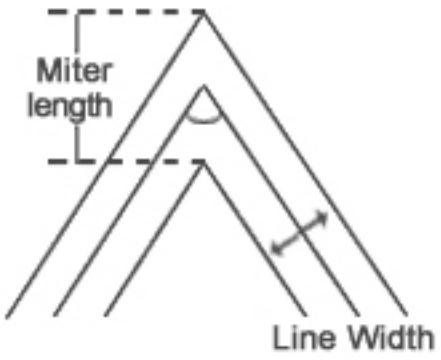
Semantics:

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

Properties:

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

Sub-elements: N/A



2.6.2.10. FILL_RULE

Semantics: set the current fill rules

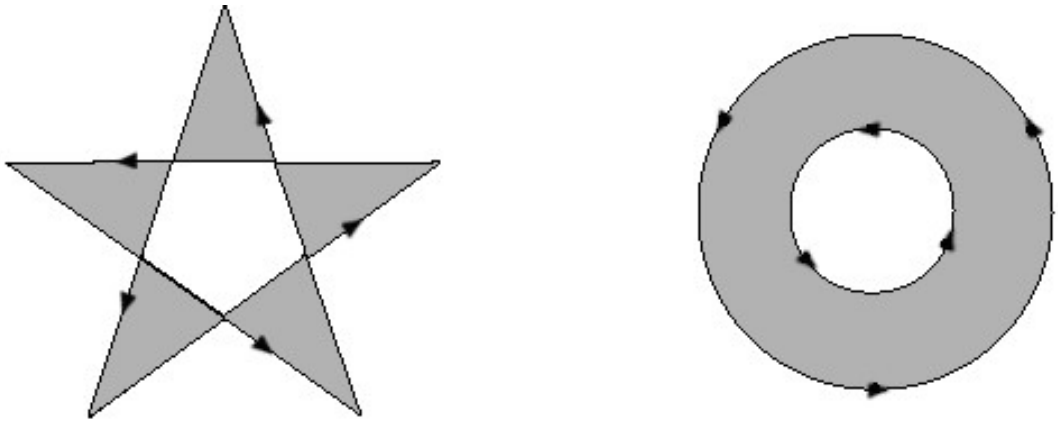
Properties:

v1: a character string, representing fill rules. Its possible value includes RULE_EVENODD and RULE_WINDING.

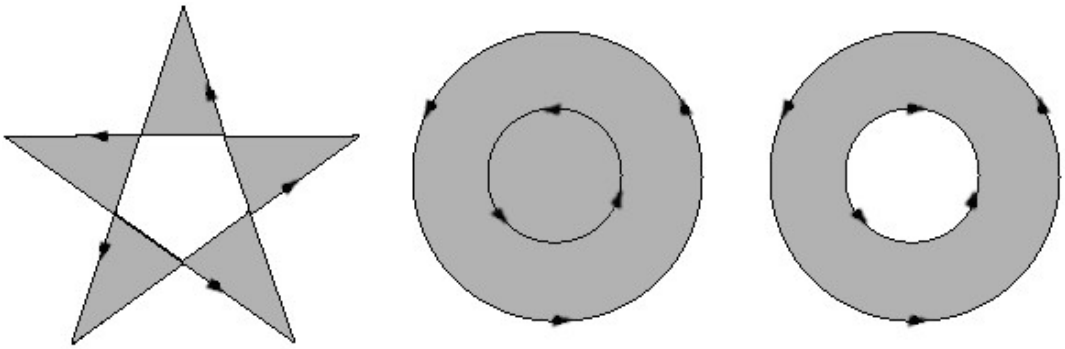
Sub-elements: N/A

Note:

RULE_EVENODD: Specifies that areas are filled according to the even-odd parity rule. According to this rule, you can determine whether a test point is inside or outside a closed curve as follows: Draw a line from the test point to a point that is distant from the curve. If that line crosses the curve an odd number of times, the test point is inside the curve; otherwise, the test point is outside the curve.



RULE_WINDING: Specifies that areas are filled according to the nonzero winding rule. According to this rule, you can determine whether a test point is inside or outside a closed curve as follows: Draw a line from a test point to a point that is distant from the curve. Starting with a count of 0, the rule adds 1 each time a curve segment crosses the ray from left to right and subtracts 1 each time a segment crosses from right to left. After counting all the crossings, if the result is 0, the point is outside the path; otherwise, it is inside.



2.6.2.11. RENDER_MODE

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

Properties:

v1: a character string, representing render mode. Its possible value includes LINE, FILL, CLIP, or some of them connected by comma.

Sub-elements: N/A

Note:

LINE: draw a line along the path..

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

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

2.6.2.12. RASTER_OP

Semantics: set the current raster operation.

Properties:

v1: a character string, representing raster operation. Its possible value includes 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.

Sub-elements: N/A

Note:

ROP_COPY: pixel_color = src

ROP_N_COPY: pixel_color = ~src

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

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

ROP_NOP: pixel_color = dest

ROP_REV: pixel_color = ~dest

ROP_AND: pixel_color = src & dest

ROP_AND_N: pixel_color = src & ~dest

ROP_N_AND: pixel_color = ~src & dest

ROP_N_AND_N: pixel_color = ~src & ~dest

ROP_OR: pixel_color = src | dest

ROP_OR_N: pixel_color = src | ~d

ROP_N_OR: pixel_color = ~src | dest

ROP_N_OR_N: pixel_color = ~src | ~dest

ROP_XOR: pixel_color = src ^ dest

ROP_EOR: pixel_color = src ^ ~dest

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

2.6.2.13. TEXT_DIR

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

Properties:

v1: a character string, representing the text direction. The possible values include HEAD_LEFT, HEAD_RIGHT, HEAD_TOP and HEAD_BOTTOM.

Sub-elements: N/A

Note:

HEAD_LEFT: the text direction is from left to right.

HEAD_RIGHT: the text direction is from right to left.

HEAD_TOP: the text direction is from top to bottom.

HEAD_BOTTOM: the text direction is from bottom to top.

2.6.2.14. CHAR_DIR

Semantics: set the current character direction, the direction is from the bottom to the top of the character.

Properties:

v1: a character string, representing character direction. The possible values include HEAD_LEFT, HEAD_RIGHT, HEAD_TOP and HEAD_BOTTOM.

Sub-elements: N/A

Note:

HEAD_LEFT: the character is heading left.

HEAD_RIGHT: the character is heading right.

HEAD_TOP: the character is heading up.

HEAD_BOTTOM: the character is heading down.

2.6.2.15. CHAR_ROTATE

Semantics: set the current character rotation angle.

Properties:

v1: a floating point number, representing the character rotating radian. Positive value means anti-clockwise; negative value means clockwise.

v2: a character string, representing whether the rotation is around the character center or around the top-left corner. The possible value includes ROT_CENTER and ROT_LEFTTOP.

Sub-elements: N/A

2.6.2.16. CHAR_SLANT

Semantics: set slant of the current character.

Properties:

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

Sub-elements: N/A

2.6.2.17. CHAR_SIZE

Semantics: set the current character width and height.

Properties:

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

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

Sub-elements: N/A

2.6.2.18. CHAR_WEIGHT

Semantics: set the current character weight.

Properties:

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

Sub-elements: N/A

2.6.2.19. CHAR_STYLE

Semantics: set the current character style.

Properties:

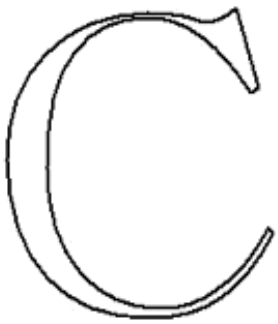
v1: a character string, representing the character style. The possible values include SHADOW, HOLLOW and OUTLINE, or some of them connected with commas. If the string is empty, that means to clean the old setting.

Sub-elements: N/A

Note:

SHADOW: set shadow style.

HOLLOW: set hollow style

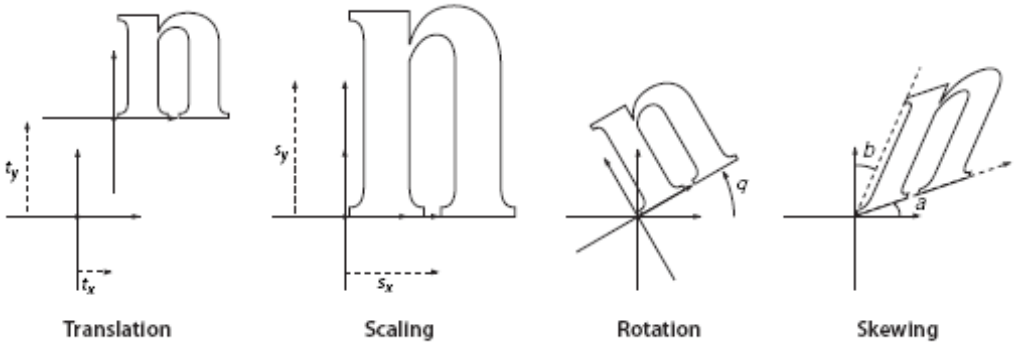


OUTLINE: set outline style.

2.6.2.20. TEXT_MATRIX

Semantics:

set the current text transformation matrix. The visual effect of transforming a character is shown below:



Properties: N/A

Sub-elements:

matrix: element of MATRIX type, responsible for transforming coordinates of text.

2.6.2.21. IMAGE_MATRIX

Semantics: set the current image transformation matrix

Properties: N/A

Sub-elements:

matrix: element of MATRIX type, used for transforming coordinates of image.

2.6.2.22. GRAPH_MATRIX

Semantics: set the current line/curve transformation matrix

Properties: N/A

Sub-elements:

matrix: element of MATRIX type, used for transforming coordinates of path graphics, such as line, Bezier curve, arc, circle, ellipse, rect, roundrect, sub-path, path, etc.

2.6.2.23. EXT_MATRIX

Semantics: set the current extension transformation matrix

Properties: N/A

Sub-elements:

matrix: element of MATRIX type, used for transforming coordinates of all path graphics, images and texts (after dedicated transformation matrix for path graphics, images and texts).

2.6.2.24. PUSH_GS

Semantics: put the current graphics state onto stack.

Properties: N/A

Sub-elements: N/A

2.6.2.25. POP_GS

Semantics: pop out the top value from the graphics state stack, replace current graphics state

Properties: N/A

Sub-elements: N/A

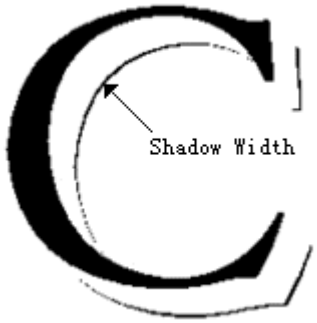
2.6.2.26. SHADOW_WIDTH

Semantics: set the border width of the current character shadow.

Properties:

v1: a floating point number, representing shadow border width.

Sub-elements: N/A



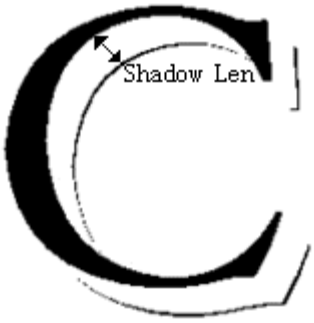
2.6.2.27. SHADOW_LEN

Semantics: set the length of the current character shadow

Properties:

v1: floating point number, representing the character shadow length.

Sub-elements: N/A



2.6.2.28. SHADOW_DIR

Semantics: set the direction of the current character shadow

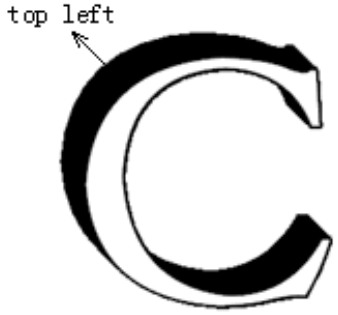
Properties:

v1: a character string. The possible values include SHADOW_LT, SHADOW_LB, SHADOW_RT and SHADOW_RB.

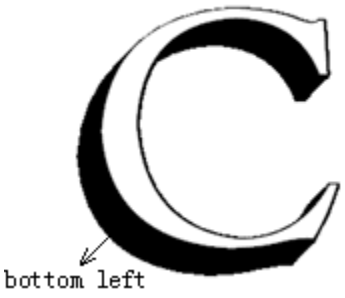
Sub-elements: N/A

Note:

SHADOW_LT: the character shadow direction is top left.



SHADOW_LB: the character shadow direction is bottom left.



SHADOW_RT: the character shadow direction is top right.

SHADOW_RB: the character shadow direction is bottom right.

2.6.2.29. SHADOW_ATL

Semantics: set whether to adjust the coordinates of a character when the direction of character shadow is to the left or top

Properties:

v1: a boolean value, representing whether to alter the coordinates of a character

Sub-elements: N/A

2.6.2.30. SHADOW_NEG

Semantics: set the current shadow character as an intaglio character

Properties:

v1: a boolean value, representing whether it is an intaglio character

Sub-elements: N/A

2.6.2.31. CLIP_AREA

Semantics: set the current clip area

Properties: N/A

Sub-elements:

cliparea: PATH type, representing the clip area

2.6.2.32. FONT

Semantics: set the font used by some encoding/character set, for example set English character to use "Arial" font.

Properties:

v1: a character string, representing the encoding/character set.

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

Sub-elements: N/A

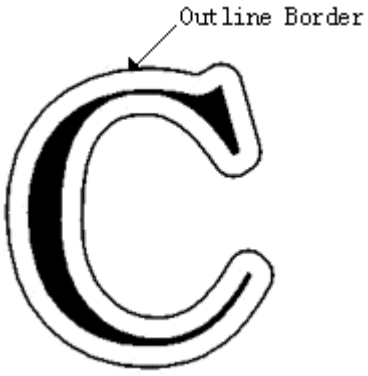
2.6.2.33. OUTLINE_BORDER

Semantics: set the border width of the current outline character

Properties:

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

Sub-elements: N/A



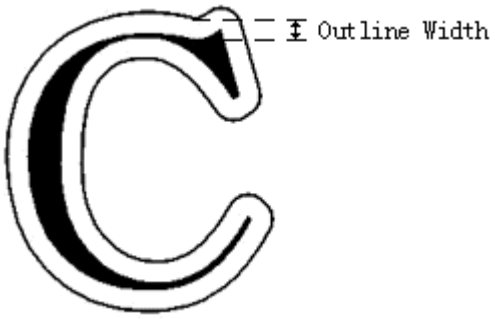
2.6.2.34. OUTLINE_WIDTH

Semantics: set the outline width of the current outline character

Properties:

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

Sub-elements: N/A



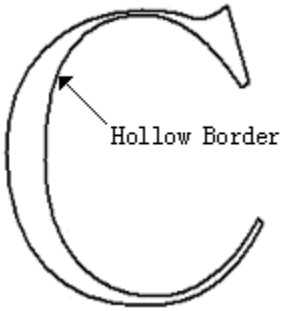
2.6.2.35. HOLLOW_BORDER

Semantics: set the border width of the current hollow character

Properties:

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

Sub-elements: N/A



2.7. Default Value of Graphics State

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

3. UOML Instructions

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

- OPEN
- CLOSE
- USE
- GET
- SET
- INSERT
- DELETE
- SYSTEM
- RET

3.1. OPEN

Semantics:

OPEN creates or opens a Docbase.

Properties:

create: a boolean value, representing whether to create a Docbase if it does not exist, default value is "true".

del_exist: a boolean value, representing whether to delete the Docbase if it already exists, default value is "false"..

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

Sub-elements: N/A

Return value:

If it succeeds, the returned RET element contains a 'stringVal' sub-element with the 'name' property as HANDLE and the 'val' property to represent the handle of the Docbase.

If it fails, the return value is defined by RET.

EXAMPLE: create a docbase, named 1.sep

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

3.2. CLOSE

Semantics:

CLOSE closes a Docbase

Properties:

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

Sub-elements: N/A

Return value:

Defined by RET

EXAMPLE: close a docbase.

<uoml:CLOSE handle="db_handle_xxxxx"/>

3.3. USE

Semantics:

USE sets an object as the current object

Properties:

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

Sub-elements: N/A

Return value:

Defined by RET

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

<uoml:USE handle="obj_handle_xxxxxx"/>

3.4. GET

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>
```

3.5. SET

Semantics:

SET property values for object. It may contain one to multiple sub-element(s) listed below.

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

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

Properties:

handle: a character string value, representing the handle, of whose property value needs to be modified. It is optional, if not exists, use the handle set from USE instead.

Sub-element:

intVal: set up integer type value, INT type

floatVal: set up float type value, DOUBLE type.

timeVal: set up time value, TIME type.

dateVal: set up date value, DATE type.

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

durationVal: set up time duration value, DURATION type.

stringVal: set up string type value, STRING type.

binaryVal: set up binary type value, BINARY type.

compoundVal: set up compound type value, COMPOUND type.

boolVal: set up boolean type value, BOOLEAN type.

Return value:

defined by RET.

EXAMPLE: set specific object's angle property.

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

3.6. INSERT

Semantics:

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

Properties:

handle: a character string value, representing the handle of parent-object. It is optional. If not exist, use the handle set from USE instead.

pos: int value, representing the inserting place, starts from 0. It is optional. If not exist, insert after the last sub-object.

Sub-element:

*xob*j: xml expression of the sub-object.

Return value:

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

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>
```

3.7. DELETE

Semantics:

DELETE deletes an object.

Properties:

handle: a character string value, representing the object to be deleted. It is optional, if not exist, use the handle set from USE instead.

Sub-element: N/A

Return value:

Defined by RET

EXAMPLE: delete an object

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

3.8. 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

Properties:

N/A

Sub-element:

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

Return value:

Defined by RET

EXAMPLE: save the Docbase example.sep

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

3.9. RET

Semantics:

RET is the return value from DCMS to application software, which may contain one to multiple return values and each value is represented by one sub-element(which can be boolVal, stringVal, intVal, floatVal, compountVal etc.).

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

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

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

For example: <boolVal name="SUCCESS" val="true"/>

Properties: N/A

Sub-element:

intVal: integer type return value, INT type

floatVal: float type return value, DOUBLE type.

TimeVal: time type return value, TIME type.

DateVal: date type return value, DATE type.

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

DurationVal: time duration type return value, DURATION type.

StringVal: string type return value, STRING type.

BinaryVal: binary type return value, BINARY type.

CompoundVal: compound type return value, COMPOUND type.

BoolVal: boolean type return value, BOOLEAN type.

Note:

RET contains at least one 'boolVal' sub-element to describe whether the operation is successful or not. Its 'name' property is SUCCESS, and its 'val' property describes whether it is successful.

When the operation fails, RET also contains one 'stringVal' sub-element. Its 'name' property is ERR_INFO, and its 'val' property describes the failure information; for other return value, check the definition of concrete instruction for reference.

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>
```

3.10.Definition of Referenced Type

3.10.1. INT

Properties:

name: a character string value, xs:string type

val: xs:integer type

Sub-element: N/A

3.10.2. DOUBLE

Properties:

name: a character string, xs:string type

val: xs:double type

Sub-element: N/A

3.10.3. LONG

Properties:

name: a character string, xs:string type

val: xs:long type

Sub-element: N/A

3.10.4. DATE

Properties:

name: a character string, xs:string type

val: xs:date type

Sub-element*:* N/A

3.10.5. TIME

Properties*:*

name: a character string, xs:string type

val: xs:time type

Sub-element*:* N/A

3.10.6. DATETIME

Properties:

name: a character string, xs:string type

val: xs:datetime type

Sub-element: N/A

3.10.7. DURATION

Properties:

name: a character string, xs:string type

val: xs:duration type

Sub-element: N/A

3.10.8. STRING

Properties:

name: a character string, xs:string type

val: xs:string type

Sub-element: N/A

3.10.9. BINARY

Properties:

name: a character string, xs:string type

val: xs:base64Binary type

Sub-element: N/A

3.10.10. BOOL

Properties:

name: a character string, xs:string type

val: xs:boolean type

Sub-element: N/A

3.10.11. COMPOUND

Property:

name: a character string, xs:string type

Sub-element:

arc: ARC type

bezier: BEZIER type

circle: CIRCLE type

cmd: CMD type

rgb: COLOR_RGB type

doc: DOC type

docbase: DOCBASE type

docset: DOCSET type

ellipse: ELLIPSE type

embedfont: EMBEDFONT type

fontlist: FONTLIST type

fontmap: FONTMAP type

image: IMAGE type

layer: LAYER type

line: LINE type

matrix: MATRIX type

meta: META type

metalist: METALIST type

page: PAGE type

path: PATH type

rect: RECT type

roundrect: ROUNDRECT type

subpath: SUBPATH type

text: TEXT type

objstream: OBJSTREAM type

Note:

Each sub-element may occur 0 to multiple times

4. Conformance

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.

Appendix A. Acknowledgments

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

Participants:

Alex Wang, Sursen Corporation

Xu Guo, Sursen Corporation

Yan Allison Shi, Sursen Corporation

Liwei Wang, Sursen Corporation

Stephen Green, Individual

Charles H. Schulz, Ars Aperta

Pine Zhang, UOML Alliance

Mendy Liu, UOML Alliance

Andy Li, Changfeng Open Standards Platform Software Alliance

Lin Cheng, Beijing Redflag CH2000 Software Co. Ltd.