WebCGM Version 2.1

Committee Draft 03
27 May 2009

Specification URIs:

This Version:

XHTML multi-file: http://docs.oasis-open.org/webcgm/v2.1/cd03/webcgm-v2.1-index.html (AUTHORITATIVE)
PDF: http://docs.oasis-open.org/webcgm/v2.1/cd03/webcgm-v2.1.pdf
XHTML ZIP archive: http://docs.oasis-open.org/webcgm/v2.1/cd03/webcgm-v2.1.zip

Previous Version:

XHTML multi-file: http://docs.oasis-open.org/webcgm/v2.1/cs01/webcgm-v2.1-index.html (AUTHORITATIVE)
PDF: http://docs.oasis-open.org/webcgm/v2.1/cs01/webcgm-v2.1.pdf
XHTML ZIP archive: http://docs.oasis-open.org/webcgm/v2.1/cs01/webcgm-v2.1.zip

Latest Version:

XHTML multi-file: http://docs.oasis-open.org/webcgm/v2.1/latest/webcgm-v2.1-index.html
PDF: http://docs.oasis-open.org/webcgm/v2.1/latest/webcgm-v2.1.pdf
XHTML ZIP archive: http://docs.oasis-open.org/webcgm/v2.1/latest/webcgm-v2.1.zip

Declared XML namespaces:

http://www.cgmopen.org/schema/webcgm/

System Identifier:

http://docs.oasis-open.org/webcgm/v2.1/webcgm21.dtd

Technical Committee:

OASIS CGM Open WebCGM TC

Chair(s):

David Cruikshank, The Boeing Company

Editor(s):

Benoit Bezaire, PTC

David Cruikshank, The Boeing Company
Lofton Henderson, Individual

Related Work:

This specification updates:
WebCGM 2.0 OASIS Standard (and W3C Recommendation)
This specification, when completed, will be identical in technical content to:
WebCGM 2.1 W3C Recommendation, available at http://www.w3.org/TR/...(tbd).../ .

Abstract:

Computer Graphics Metafile (CGM) is an ISO standard, defined by ISO/IEC 8632:1999, for the interchange of 2D vector and mixed vector/raster graphics. WebCGM is a profile of CGM, which adds Web linking and is optimized for Web applications in technical illustration, electronic documentation, geophysical data visualization, and similar fields. First published (1.0) in 1999, WebCGM unifies potentially diverse approaches to CGM utilization in Web document applications. It therefore represents a significant interoperability agreement amongst major users and implementers of the ISO CGM standard.

The present version, WebCGM 2.1, refines and completes the features of the major WebCGM 2.0 release. WebCGM 2.0 added a DOM (API) specification for programmatic access to WebCGM objects, a specification of an XML Companion File (XCF) architecture, and extended the graphical and intelligent content of WebCGM 1.0.

The design criteria for WebCGM aim at a balance between graphical expressive power on the one hand, and simplicity and implementability on the other. A small but powerful set of standardized metadata elements supports the functionalities of hyperlinking and document navigation, picture structuring and layering, and enabling search and query of WebCGM picture content.

Status:

This document was last revised or approved by the OASIS CGM Open WebCGM TC on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

The final version of this specification is to be produced jointly by OASIS and W3C and published simultaneously as an OASIS Standard and a W3C Recommendation. The two documents will have identical content except for cover page and formatting differences as appropriate to the two organizations.

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 www.oasis-open.org/committees/tc_home.php?wg_abbrev=cgmo-webcgm .)

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 (www.oasis-open.org/committees/cgmo-webcgm/ipr.php .)

The non-normative errata page for this specification is located at www.oasis-open.org/committees/cgmo-webcgm.

Notices

Table of Contents

Up to cover page

WebCGM 2.1 — Expanded Table of Contents

Contents

WebCGM 2.1 — Introduction to WebCGM

---

1. Introduction to WebCGM

This chapter's sections are informative, unless otherwise indicated.

Contents

1.1 Terminology

This section is normative.

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

1.2 Normative references

This section is normative.

ISO/IEC 8632:1999(E)

Information technology - Computer graphics - Metafile for the storage and transfer of picture description information
- Part 1: Functional description
- Part 3: Binary encoding
- Part 4: Clear text encoding

Available at the ISO page of Publicly Available Standards. CGM:1999 was reaffirmed by ISO, without changes, at its 5-year review in 2004. The WebCGM profile is defined by reference to the ISO standard.

ISO Register

ISO International Register of Graphical Items, the normative repository of registered extensions to ISO CGM. Available at http://jitc.fhu.disa.mil/nitf/graph_reg/graph_reg.html. An informative summary of registered CGM items, including pointers into the normative register, is available at http://www.cgmopen.org/technical/registry/ .

ISO/IEC 8632-1:1999/Cor 1:2006

This corrigendum to CGM:1999 corrects an error in the specification of NURBS knots list in CGM:1999. Available at: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44114 .

ISO/IEC 8632-1:1999/Cor 2:2007

This corrigendum to CGM:1999 clarifies the permissibility and usage rules of Application Structures in the Text Open State of CGM:1999. Available at: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50315 .

RFC 3986

Uniform Resource Identifiers (URI): Generic Syntax, Eds. T.Berners-Lee, R.Fielding, L.Masinter, January 2005, available at http://www.ietf.org/rfc/rfc3986.txt .

RFC 3987

Internationalized Resource Identifiers (IRIs), M.Duerst, M. Suignard, January 2005, available at http://www.ietf.org/rfc/rfc3987.txt .

RFC 1951

Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC1951, Aladdin Enterprises, May 1996. Available at: http://www.ietf.org/rfc/rfc1951.txt .

RFC 1952

Deutsch, P., "GZIP file format specification version 4.3", RFC1952, Aladdin Enterprises, May 1996, URL: http://www.ietf.org/rfc/rfc1952.txt .

ISO/IEC 10646

ISO (International Organization for Standardization). *ISO/IEC 10646-1:2000. Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane* and *ISO/IEC 10646-2:2001. Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 2: Supplementary Planes*, as, from time to time, amended, replaced by a new edition or expanded by the addition of new parts. [Geneva]: International Organization for Standardization. (See http://www.iso.ch for the latest version.)

ISO/IEC 10646-UTF8

ISO/IEC 10646-1:1993, AM2:1996, Information technology — Universal multiple-octet coded

character set (UCS) — Part 1: Architecture and Basic Multilingual Plane AMENDMENT 2: UCS Transformation Format 8 (UTF-8). Available from ISO, see http://www.iso.ch .

ITU-jpeg

Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines. ITU-T Recommendation T.81. Available at http://www.w3.org/Graphics/JPEG/itu-t81.pdf. . (ISO/IEC 10918-1 : 1993(E))

REC-png

Portable Network Graphics (PNG) Specification (Second Edition) - Information technology - Computer Graphics and image processing - Portable Network Graphics (PNG): Functional specification. ISO/IEC 15948:2003(E). W3C Recommendation 10 November 2003, available at http://www.w3.org/TR/2003/REC-PNG-20031110 .

RFC 2119

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

XML 1.0

XML 1.0, fourth edition, Eds. T.Bray, J.Paoli, C.M.Sperberg-McQueen, E.Maler, F.Yergeau, August 2006, available at http://www.w3.org/TR/2006/REC-xml-20060816/ .

Namespaces in XML

Namespaces in XML 1.0 (Second Edition), Eds. T.Bray, D.Hollander, A.Layman, R.Tobin, August 2006, available at http://www.w3.org/TR/2006/REC-xml-names-20060816 .

RFC 2781

IETF (Internet Engineering Task Force). *RFC 2781: UTF-16, an encoding of ISO 10646*, Eds. P. Hoffman, F. Yergeau., February 2000. (Available at http://www.ietf.org/rfc/rfc2781.txt)

RFC 3629

UTF-8, a transformation format of ISO 10646, IETF RFC 3629, STD 63, Ed. F. Yergeau, November 2003. (See http://www.ietf.org/rfc/rfc3629.txt)

Unicode

The Unicode Consortium, *The Unicode Standard, Version 4*, ISBN 0-321-18578-1, as updated from time to time by the publication of new versions. (See http://www.unicode.org/unicode/standard/versions for the latest version and additional information on versions of the standard and of the Unicode Character Database).

1.3 Non-normative references

SVG 1.1

Scalable Vector Graphics (SVG) 1.1 Specification, Eds. J.Ferraiolo, J.Fujisawa, D.Jackson, January 2003, available at http://www.w3.org/TR/SVG11/ .

DOM Level 3 Core

Document Object Model (DOM) Level 3 Core Specification, Eds. A.Le Hors, P.Le Hégaret (plus L. Wood, G.Nicol, J.Robie, M.Champion, S.Byrne), April 2004, available at http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/ .

DOM Level 3 Events

Document Object Model Level 3 Events Specification, Eds. B.Höhrmann, P.Le Hégaret, T.Pixley, April 2006, available at http://www.w3.org/TR/2003/NOTE-DOM-Level-3-Events-20031107.

HTML 4.01

HTML 4.01 Specification, Eds. D.Raggett, A.Le Hors, I.Jacobs, December 1999, available at http://www.w3.org/TR/html401/ .

CSS 2.0

Cascading Style Sheets, level 2, CSS2 Specification, Eds, B.Bos, H.Wium Lie, C.Lilley, I.Jacobs, May 1998, available at http://www.w3.org/TR/CSS2/ .

Xpointer Framework

XPointer Framework, Eds. P.Grosso, E.Maler, J.Marsh, N.Walsh, March 2003, available at http://www.w3.org/TR/2003/REC-xptr-framework-20030325/ .

Cascading Profiles

Definition and description of how to write a profile using WebCGM as the starting point, for closely related technical application sectors. At http://www.cgmopen.org/technical/cascading-profiles.html .

WebCGM 2.1 Requirements

The requirements used to define the new functionality for WebCGM 2.1. At http://docs.oasis-open.org/webcgm/v2.1/WebCGM_21_Requirements.html.

WebCGM 2.0 Requirements

The requirements on which the major WebCGM 2.0 release was based. At http://www.cgmopen.org/technical/WebCGM_20_Requirements.html .

SpecGL 1.0

The W3C *QA Framework: Specification Guidelines* has guided the inclusion of the normative Conformance clause, and other conformance-related details of WebCGM. Eds. K.Dubost, L. Rosenthal, D.Hazaël-Massieux, L.Henderson,August 2005, available at http://www.w3.org/TR/qaframe-spec/ .

UAAG 1.0

User Agent Accessibility Guidelines 1.0, Eds. Ian Jacobs, Jon Gunderson, Eric Hansen, 17 December 2002, a W3C Recommendation available at http://www.w3.org/TR/2002/REC-UAAG10-20021217/ .

WCAG 1.0

Web Content Accessibility Guidelines 1.0, Eds. Wendy Chisholm, Gregg Vanderheiden, Ian Jacobs, 5 May 1999, a W3C Recommendation available at http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/ .

1.4 About WebCGM

The scope of this WebCGM™ 2.1 specification includes three components.

1. an intelligent graphics profile of the ISO Computer Graphics Metafile (CGM) standard (ISO/IEC 8632:1999), tailored to the requirements for scalable 2D vector graphics in electronic documents on the World Wide Web;
2. a WebCGM Document Object Model (DOM), which provides an application programming interface to WebCGM objects in WebCGM-supporting applications;
3. definition of a standard WebCGM XML Companion File (XCF), which allows applications to externalize some non-graphical metadata from WebCGM instances, yet maintain a tight binding of the metadata to WebCGM objects.

WebCGM is a set of specifications targeted especially at the effective application of the ISO CGM:1999

standard to representation of 2D graphical content within Web documents.

CGM has been an ISO standard since 1987, and CGM has been a registered media type (image/cgm) for the Internet and the World Wide Web since December 1995. WebCGM 1.0, comprising the original intelligent graphics profile of ISO CGM, was first published in 1999, was re-released in 2001 with error corrections, and formed the basis for the significant extensions of the WebCGM 2.0 specification.

For much more information about WebCGM, please see the WebCGM FAQ, as well as other numerous other references and reading materials, on the OASIS CGM Open Web site.

1.5 The WebCGM profile and profile rules

The WebCGM profile is a conforming profile of ISO CGM under the stipulations of CGM:1999 Clause 9, "Profiles and conformance", and it utilizes the Profile Proforma (PPF) of CGM:1999 Annex I.1, Proforma tables, for representation of the element-by-element content details.

The WebCGM profile is an "intelligent graphics" profile, which means that in addition to graphical content based on CGM Versions 1-3, the profile includes non-graphical content based on CGM Version 4, Application Structures. The non-graphical content allows the definition of hierarchies of application objects, as well as the association of metadata, such as link specifications and layer definitions, with the objects.

1.6 WebCGM requirements

The original WebCGM 1.0 profile resulted from a collaboration between the CGM Open Consortium and W3C Graphics Activity. The requirements that determined the content selection for WebCGM 1.0 were derived from:

- the requirements articulated in the document "W3C Scalable Graphics Requirements" *(http://www.w3.org/Graphics/ScalableReq*);
- the specifications and requirements defined in "Use of CGM as a Scalable Graphics Format" (*http://www.w3.org/TR/NOTE-cgm* [cgmreq]);
- technical recommendations from the CGM Open Consortium (see *http://www.cgmopen.org/*).

The selection criteria for the WebCGM profile include:

- graphical content: it should have high expressive power; and, it should be both widely implemented, and implementable with a reasonable level of effort.
- intelligence content (structuring and metadata elements): criteria came from the above-mentioned requirements document, [cgmreq], plus additional requirements generated during the first 5 years of deployment and use of the WebCGM 1.0 standard.

The requirements of the major WebCGM 2.0 release -- a set of additions, deletions, and modifications

applied to the 1.0 profile -- were shaped by:

- additional requirements generated during 5 years of deployment of WebCGM 1.0 in industry;
- apparent non-use of certain 1.0 features;
- need for convergence with similar profiles in closely related industries.

The content of the WebCGM 2.1 profile comprises less than a dozen items that were arguably within the scope of WebCGM 2.0, but which arose too late in the standardization of the latter. The WebCGM 2.1 Requirements document summarizes these requirements.

1.7 WebCGM and other profiles

The WebCGM 2.1 intelligent graphics profile, like its predecessors WebCGM 2.0 and WebCGM 1.0, is a profile of the ISO CGM:1999 standard, designed for effective application of CGM in technical Web applications. WebCGM is not aimed at or optimized for any particular technical application sector, but is intended to satisfy general requirements shared by different but closely related technical Web applications.

Following five years of deployment and application of WebCGM and other technical profiles (such as Air Transport Association's), some divergence began to appear. WebCGM 2.0 represented a major effort towards convergence of intelligent graphics profiles in closely related industries. In fact, starting with version 2.0, it is the intention of the authors and publishers of WebCGM that it be used as a basis for the definition of industry-specific profiles. Cascading Profiles describes the use of WebCGM as a core profile from which specific industries derive and define their technical profiles.

1.8 Editions and releases

CGM:1999 Clause 9, "Profiles and conformance", prescribes that profiles shall maintain revision control by using a standard "ProfileEd" keyword. Instances of a profile carry this edition information in their identification section. Prior releases of WebCGM include:

- *WebCGM Profile*, 21 January 1999, the first release of WebCGM ("ProfileEd:1.0").
- *WebCGM 1.0 Second Release*, 17 December 2001, an errata-correcting release of the previous (also "ProfileEd:1.0").
- *WebCGM 2.0*, a major functional upgrade of WebCGM 1.0, simultaneously published by OASIS and W3C on 30 January 2007 ("ProfileEd:2.0").

This specification is the first release of WebCGM 2.1 ("ProfileEd:2.1"). There may be future releases of WebCGM 2.1, for maintenance and defect correction. There may be future higher editions and versions of WebCGM (e.g., 2.X).

1.9 Roadmap to this specification

WebCGM is written in these major sections:

- This section, containing introductory and overview material, which is *mostly* informative, but does contain two *normative* subsections.
- A WebCGM Concepts section, *informative* but *not normative*.
- Detailed descriptive material on the V4 content of WebCGM, including Content Model which can be used for V4 content. This section is *normative*.
- The definition of a standard XML Companion File (XCF) for use with WebCGM. This section is *normative*.
- The definition of the WebCGM Document Object Model (DOM). This section is *normative* (including normative IDL definitions of the DOM interfaces).
- The Profile Proforma (PPF), comprising an extensive table which addresses every element of the ISO CGM standard, per CGM:1999 Annex I. This section is *normative*.
- The normative Conformance chapter describes the conformance targets and conformance details of WebCGM.
- The ECMAScript chapter give a normative description of an ECMAScript binding for WebCGM DOM.
- The Application Configurable Items chapter defines an XML format for defaults specification on a number of WebCGM elements, and for font substitution.
- Appendixes, including informative sections such as revision history, comparison of WebCGM 2.1 with the previous version, etc.

Note about **CGM examples**. In Chapter 5, defining the WebCGM DOM, there are examples that end with text lines, "*View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)*" In document formats that support external links (i.e., XHTML), each of these examples links to an XHTML snippet that invokes WebCGM instances. To view them your browser must have a WebCGM viewer plug-in, control, or appropriate equivalent technology. To obtain such a viewer, see for example the (non-exhaustive) CGM products directory on the OASIS/CGM Open Web site.

1.10 Document sources and registration authority

Copies of the ISO standards may be obtained from ISO:

ISO Central Secretariat
International Organization for Standardization (ISO)
ch. de la Voie-Creuse
Case postale 56
CH-1211 Geneva 20
Switzerland

For the purpose of this Recommendation and according to the rules for the designation and operation of registration authorities in the ISO/IEC Directives, the ISO and IEC Councils have designated the following as the registration authority:

Joint Interoperability Test Command
ATTN: JTF NITFS Registration Authority (ISO/IEC 9973)
P.O. Box 12798
Fort Huachuca, AZ 85670-2798
USA

For more information on WebCGM and the CGM:1999 standard itself, the CGM Open Web site has a collection of bibliographic references and short articles:

CGM Open
http://www.cgmopen.org/

These additional World Wide Web sites have more information on CGM:

W3C WebCGM Overview
http://www.w3.org/Graphics/WebCGM

Back to top of chapter

WebCGM 2.1 — WebCGM Concepts

---

2. WebCGM concepts

This chapter is informative (non-normative).

Contents

2.1 The structure of a WebCGM

A WebCGM is a Version 1, 2, 3, or 4 CGM as defined in ISO/IEC 8632:1999, with some restrictions. The restrictions improve the interoperability of WebCGM, and simplify the production of WebCGM interpreter (viewer) tools.

A WebCGM 2.1 instance, as shown in Figure 1, consists of a single Picture.

Properties which apply to the whole metafile are defined in the Metafile Descriptor. These include descriptive information about the metafile, the precisions of numbers, as well as identifiers for fonts and such resources. Properties which apply to the elements in the body of the picture are contained in the Picture Descriptor. These include such information as picture size and scaling, specification modes for aspects such as line width, and background color. Because WebCGM 2.1 allows only a single picture per metafile, the distinction -- whole-metafile versus picture-specific -- may not seem useful. However, because a WebCGM 2.1 metafile must be a valid ISO CGM:1999 metafile, the ISO CGM:1999 metafile

structure is observed.

The WebCGM picture contains CGM graphic elements, as well as (optionally) Application Structures. Application Structures define intelligent objects within the picture, which are comprised of groups of graphical primitives. These intelligent objects may contain attributes or properties. WebCGM defines several types of intelligent objects - "graphical object", "paragraph", "layer", and "sub-paragraph" - as well as a few properties which each group may have.

WebCGM 2.1 also contains a purely graphical grouping mechanism, "graphical node", which groups graphical primitives as an Application Structure, but disallows the attributes or properties that associate intelligence with objects.

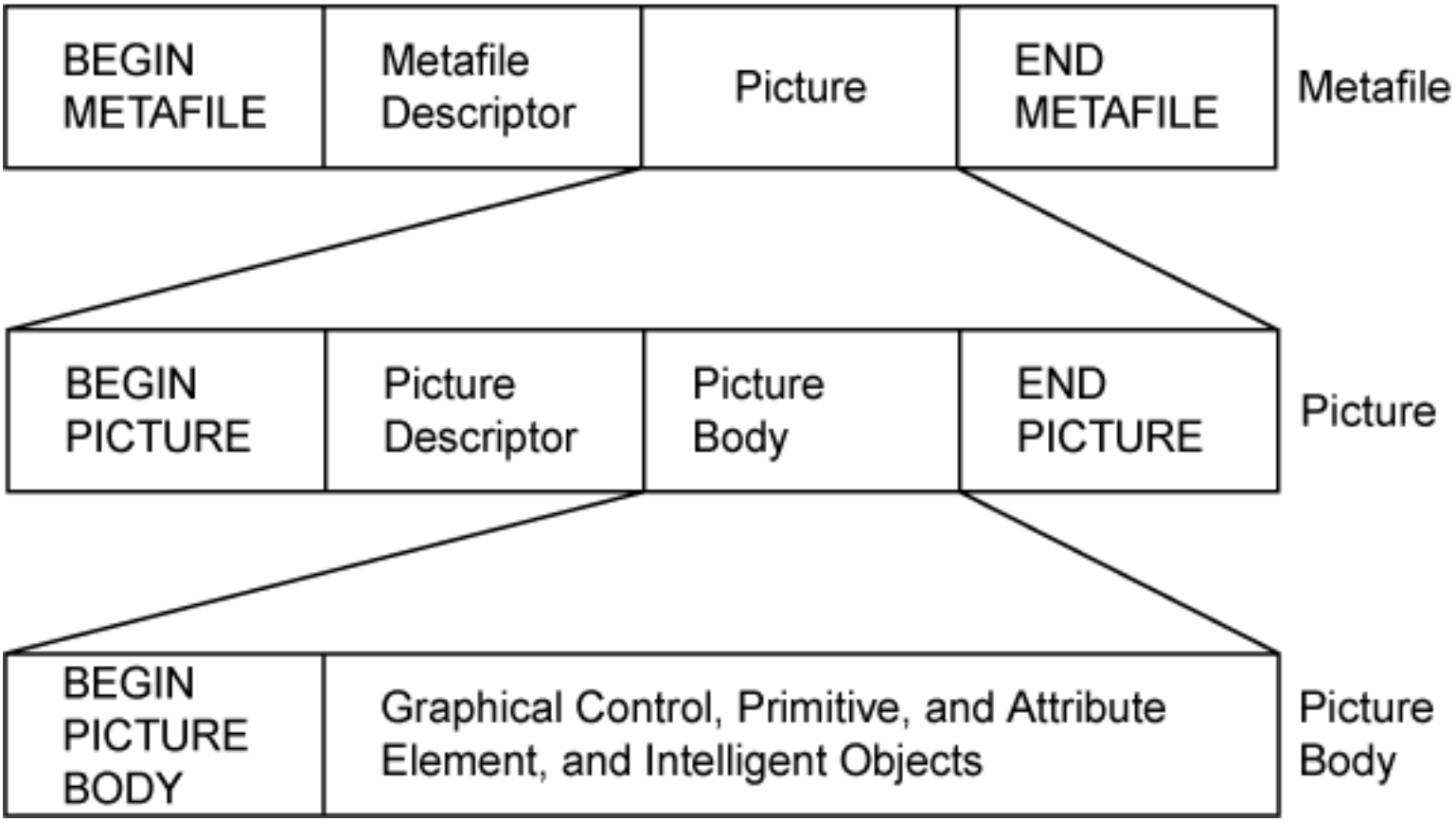

Figure 1. WebCGM File Structure

2.2 Picture content and usage

2.2.1 Raster and vector content

CGM supports both raster and vector graphics in the same picture. WebCGM permits the use of popular raster compression methods — ITU-T Group 4, JPEG, and the deflate (LZ77 derivative) method of PNG — for raster content embedded within the picture.

For information about scaling of WebCGM pictures in Web documents, see section "WebCGM and the `object` element".

2.2.2 Drawing model

This section presents an informative description of the normative drawing model of the ISO CGM standard, as well as registered extensions that have been incorporated into the WebCGM profile.

Elements rendered first may be wholly or partially hidden by elements rendered later. In the ISO CGM standard, the writing mode of primitives is "replacement mode" — content is rendered opaquely on top of previous content. To meet per-primitive (and per-pixel) transparency requirements, WebCGM includes a registered extension for Alpha transparency, as well as registered color models RGB-alpha and sRGB-alpha.

Implementations of WebCGM are expected to behave as though they implement a drawing model corresponding to the one described below. A real implementation is not required to implement the model in this way, but the result on any device supported by the implementation shall match that described by this model.

WebCGM uses a painters model of rendering. Colors are applied in successive operations to the output device. When an area overlaps a previously colored area the new color partially or completely obscures the old. When the color is not completely opaque the result on the output device is defined by the following (mathematical) rules for compositing (all color values use premultiplied alpha):

```
Pr, Pg, Pb    – Primitive color value
Pa            – Primitive alpha value
Cr, Cg, Cb    – Canvas color value (before blending)
Ca            – Canvas alpha value (before blending)
Cr', Cg', Cb' – Canvas color value (after blending)
Ca'           – Canvas alpha value (after blending)

Ca' = 1 - (1 - Pa) * (1 - Ca)
Cr' = (1 - Pa) * Cr + Pr
Cg' = (1 - Pa) * Cg + Pg
Cb' = (1 - Pa) * Cb + Pb
```

Alpha compositing is performed in the current COLOUR MODEL (see T.16.19).

Primitives in a WebCGM document have an implicit drawing order, with the first primitives in the WebCGM document getting drawn first. Subsequent primitives are drawn on top of previously drawn primitives.

Primitives which have a value for the registered Escape 45 other than fully opaque, have the effect of producing a temporary separate canvas initialized to transparent black onto which the primitive is drawn. The canvas is then composited into the background, taking into account the Escape 45 value. The

presence of APS in the primitive list has no effect on the rendering. No temporary canvas is created. It is identical to the case of no APS.

2.2.3 Overlaying a picture

In the ISO CGM:1999 standard, a picture has an implicit or explicit opaque background. Graphic elements within the picture are rendered in the order they appear in the metafile. It is a requirement of a 2D graphics format for Web documents that pictures may be overlaid on previous content. For this, it must be controllable whether the picture background is opaque or transparent (both cases are needed), or "translucent" (partially opaque).

Conceptually, a CGM picture's background is handled as follows. When a picture's canvas is first created in the compositing model of section 2.2.2, it is initialized to transparent black (0,0,0,0). Before the drawing of the first foreground primitives, the canvas is then filled per the equations in section 2.2.2 with the *effective background color* of the metafile.

In metafiles that use the RGB-alpha color model, the effective background color may be directly set in the Picture Descriptor to any valid (r,g,b,a), including transparent black (0,0,0,0). In RGB metafiles, the same effects may be achieved by including the registered Escape 45 (alpha transparency) element in the Picture Descriptor, which is then combined with the defined RGB background color to achieve any valid (r, g,b,a) effective background color.

2.3 Intelligence — Objects, Layers, Hyperlinks, Metadata

2.3.1 Overview

Within a WebCGM picture, groups of graphical primitives can be defined which structure graphics to meet the requirements of integration into Web documents. Groups in WebCGM are realized as standard Version 4 Application Structures (APS) of ISO CGM.

To meet the requirements of intelligent graphics, four specific group types are defined and allowed in WebCGM: 'grobject', 'layer', 'para', and 'subpara'. WebCGM allows a fifth group type, 'grnode', as a convenience for authoring tools to preserve their graphical grouping functions. The detailed normative syntax and semantics of the group types, including viewer behavior, is defined in Chapter 3 and in the PPF. Below is a brief conceptual summary.

Every group has at least one explicit property, its unique identifier (a parameter of the Begin APS element). WebCGM groups other than 'grnode' may have several explicit attributes associated with them. These attributes are realized as standard Version 4 (V4) Application Structure Attribute elements (APS Attributes) of ISO CGM.

Chapter 3 normatively defines the detailed content model for version 4 elements in WebCGM using EBNF notation. See section, "WebCGM Content Model", for an informative (non-normative), all-at-once

presentation of the content model using XML DTD notation.

2.3.2 WebCGM defined group types

WebCGM defines a set of allowable group (APS) types, to support the Web document operations of hyperlinking, layered pictures, and text search within graphics. See Chapter 3 for the detailed normative syntax and semantics of the allowable group types. Brief conceptual descriptions follow (each item is linked to its Chapter 3 normative definition):

- grobject — (graphical object) the basic grouping APS for identification of objects, principally used to identify sources and destinations of hyperlinks.
- layer — an APS type that allows the division of a picture into a set of graphical layers, for use by viewers in selective presentation and "2-1/2 D" effects.
- para — (paragraph) an APS type to facilitate text search within graphics, in cases such as multi-element, multi-line text, and other cases (e.g., polygonized text) where text search might otherwise be difficult (or impossible).
- subpara — may be used to identify smaller fragments of text within APS of type 'para', enabling, for example, the marking of the larger text block (the "paragraph") for searching purposes, and the tagging of smaller fragments as hotspots.

WebCGM does allow one other group type for the convenience of authoring tools:

- grnode — may be used for simple grouping of graphical primitives, to preserve authoring tools' grouping functions; none of the facilities (properties & attributes) for attaching intelligence may be used with 'grnode'.

Note that 'grnode' was not present in WebCGM 1.0, but was added to WebCGM 2.0 to allow for better hierarchical structure in WebCGM documents. The 'grnode' ("graphical node") APS allows illustration authoring tools to preserve in the WebCGM metafile instance the graphical groupings that are often used by such tools.

WebCGM does not allow private group types in WebCGM instances. External private metadata can be associated, by id or by name, with all group (APS) types other than 'grnode' within a WebCGM. A standard external mechanism is defined in the *XML Companion File* section.

2.3.3 Usage of WebCGM objects for navigation

Groups of types 'para', 'subpara', and 'grobject' may be used for picking and navigation operations in hyperlinked Web documents. These three APS types are called "objects" in WebCGM.

Objects may contain an explicit 'region' APS Attribute, which provides the boundary for picking operations. This is known as the *overlay model* of object identification (for picking, mouseover screentip display, etc). It is useful in cases where the picking region of an object can not be defined by existing geometry, for

example on line art drawings or raster content.

Objects which contain graphical content have an implicit property: the boundary or bounding extent of the enclosed graphical object. This extent is used for picking and navigation operations in hyperlinked Web documents, in the absence of a 'region' attribute. Use of this implicit boundary property for picking and navigation operations in Version 4 CGM instances is referred to as the *embedded model*.

Objects may also be the target of a link. Viewers will generally move the APS into view and scale them to fit into the viewer's rectangle. The exact viewer behavior is controlled by a set of *object behavior* keywords associated with the link, and the presence or absence of certain APS Attributes on the object ('viewcontext', 'region', etc.)

2.3.4 WebCGM defined group properties

Explicit properties or attributes of WebCGM groups are encoded as APS Attribute elements. Each APS Attribute has a "type" parameter, which identifies the attribute. See Chapter 3 for the detailed normative syntax and semantics of the allowable APS Attributes. Brief conceptual descriptions follow (each item is linked to its Chapter 3 normative definition):

- region — defines a spatial region that can be associated with an APS for picking and navigation purposes.
- viewcontext — defines a rectangle that can be associated with an APS for establishing the initial view upon execution of a link to that APS.
- linkuri — defines a link, whose target is specified by an IRI, for hyperlinking to text content, pictures in other metafiles, or objects within the same or other metafiles. "#" fragment syntax is defined for a full addressing model down to the object level within the picture, and specifying viewer behavior upon link traversal.
- layername — the name (required) to be assigned within an APS of type 'layer'.
- layerdesc — specifies a "layer description" string within an APS of type 'layer'.
- screentip — a string to be associated with an object, to be shown in the typical Web browser "screen tip" style when cursor passes over the object.
- name — a "common name" attribute to be associated with an object, that gives a useful search handle or way of defining searchable subtypes of the object type; also allows a group of same-name objects to be a link target.
- content — an attribute of the 'para' and 'subpara' APS, that can provide a basis for applications to build text search in cases that might otherwise prove difficult (e.g., displayed text that is stroked, rasterized, fragmented, or non-natural presentation order).
- visibility — the 'visibility' attribute indicates if an object is visible (drawn) or not, and also disables its eligibility to be picked (invisible objects are ineligible for picking).
- interactivity — the 'interactivity' attribute indicates whether or not an object is eligible to be picked (i.e., may receive mouse events), and also affects a handful of other interaction-related behaviors.

WebCGM does not allow private attribute types in WebCGM instances. External private metadata, including attributes as well as elements, can be associated by id or by name with all group (APS) types

other than 'grnode' within a WebCGM. A standard external metadata binding mechanism is defined in the *XML Companion File* chapter .

2.3.5 Content Model

The detailed normative syntax and semantics are presented later (Chapter 3) in this specification. The structure and relationships of the intelligent content are illustrated in the following diagrams. In the following, "picbody" is not a specific WebCGM object type, but rather a convenience to refer to that part of the CGM picture which is between the Begin Picture Body element and the End Picture Element, exclusive. Boxes with heavy borders indicate elements that are decomposed further, and offset boxes indicate attributes associated with an element. Similarly, gdata is not an object type, but rather a catch-all reference to zero or more CGM graphical elements which WebCGM allows, and which are valid at such a position according to the rules of CGM. The cgmprim attribute associated with gdata represents an entity that associates the graphical primitives to the model. See Figure 2.

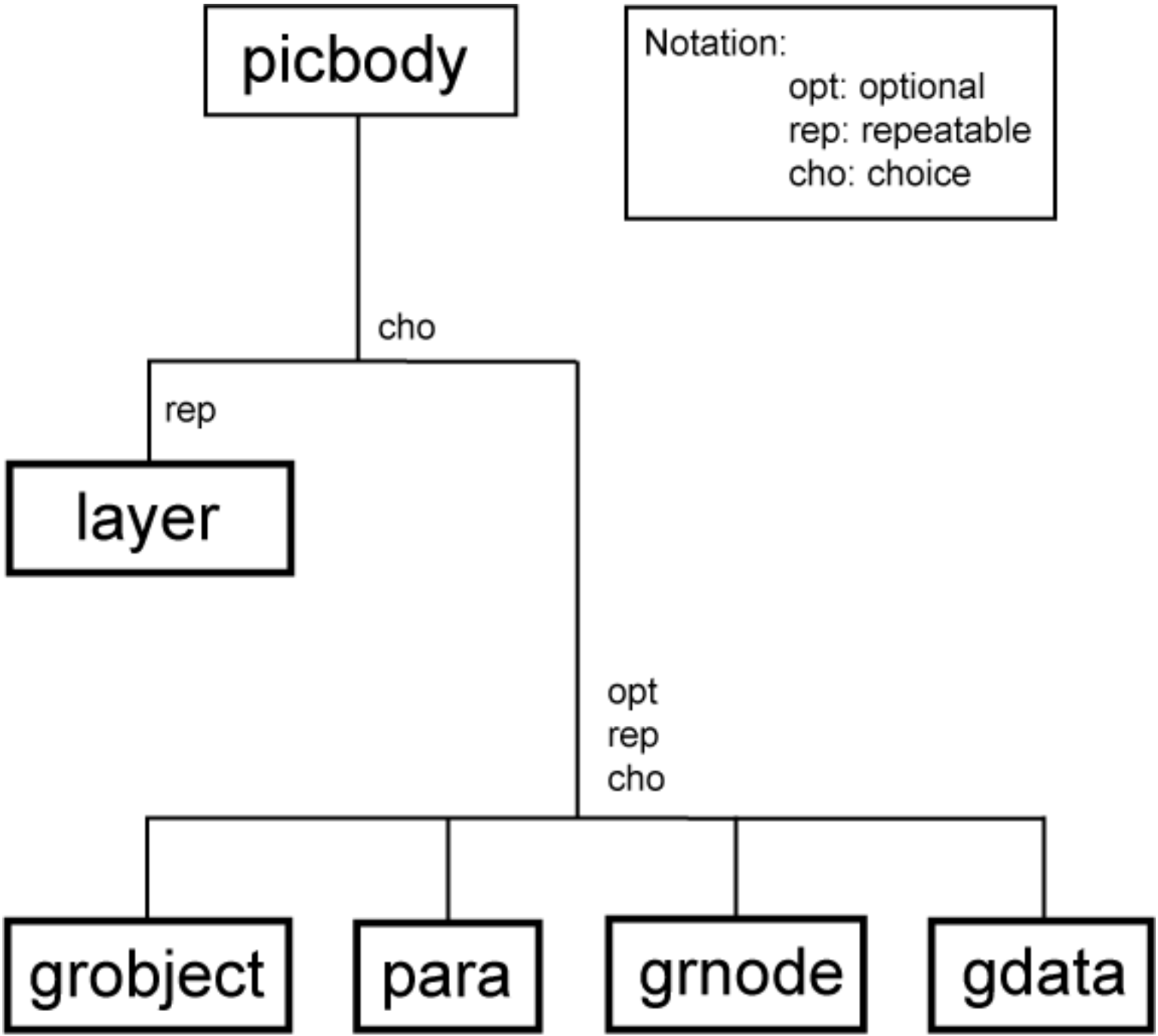

Figure 2a. WebCGM File Structure - picbody

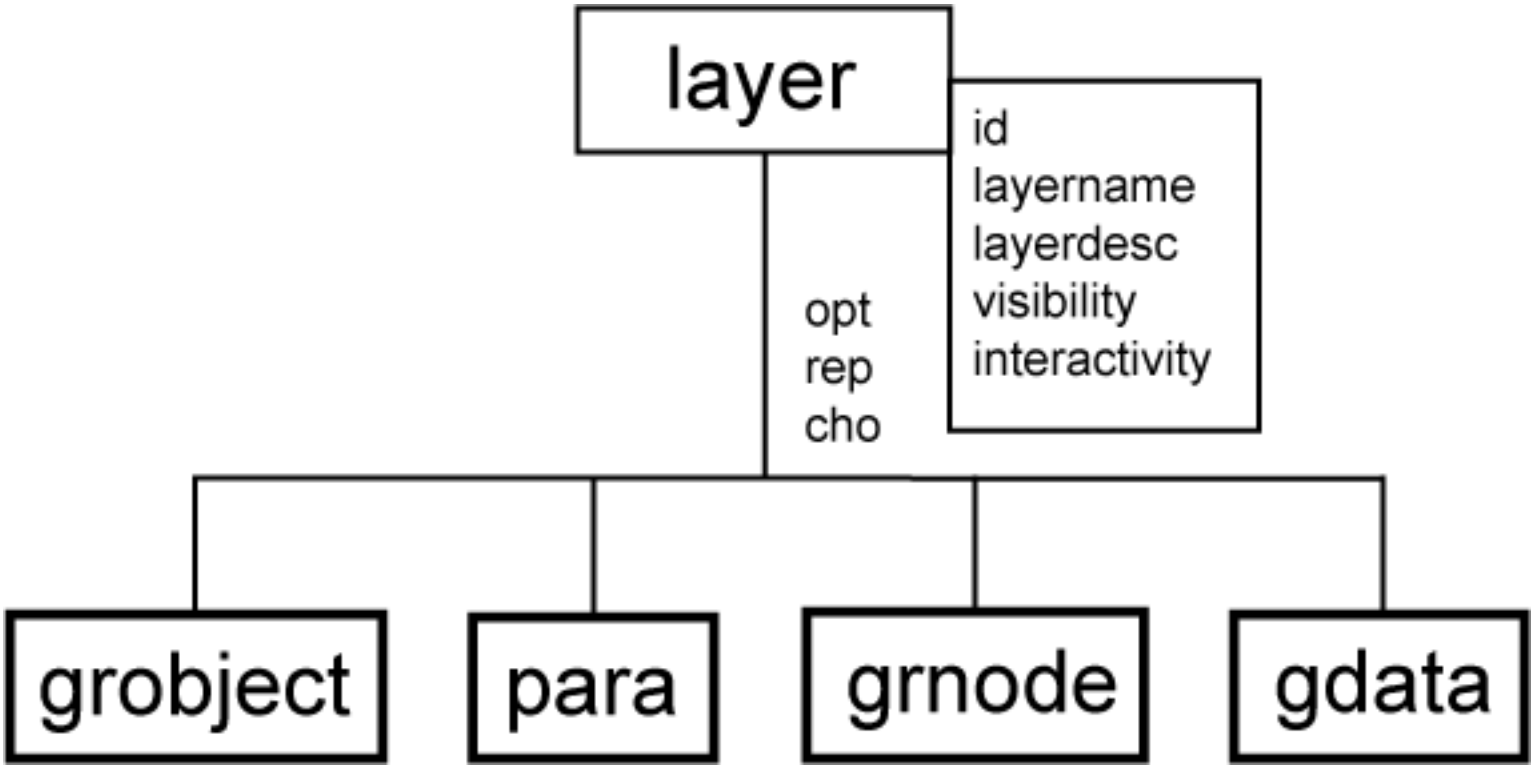

Figure 2b. WebCGM File Structure - layer

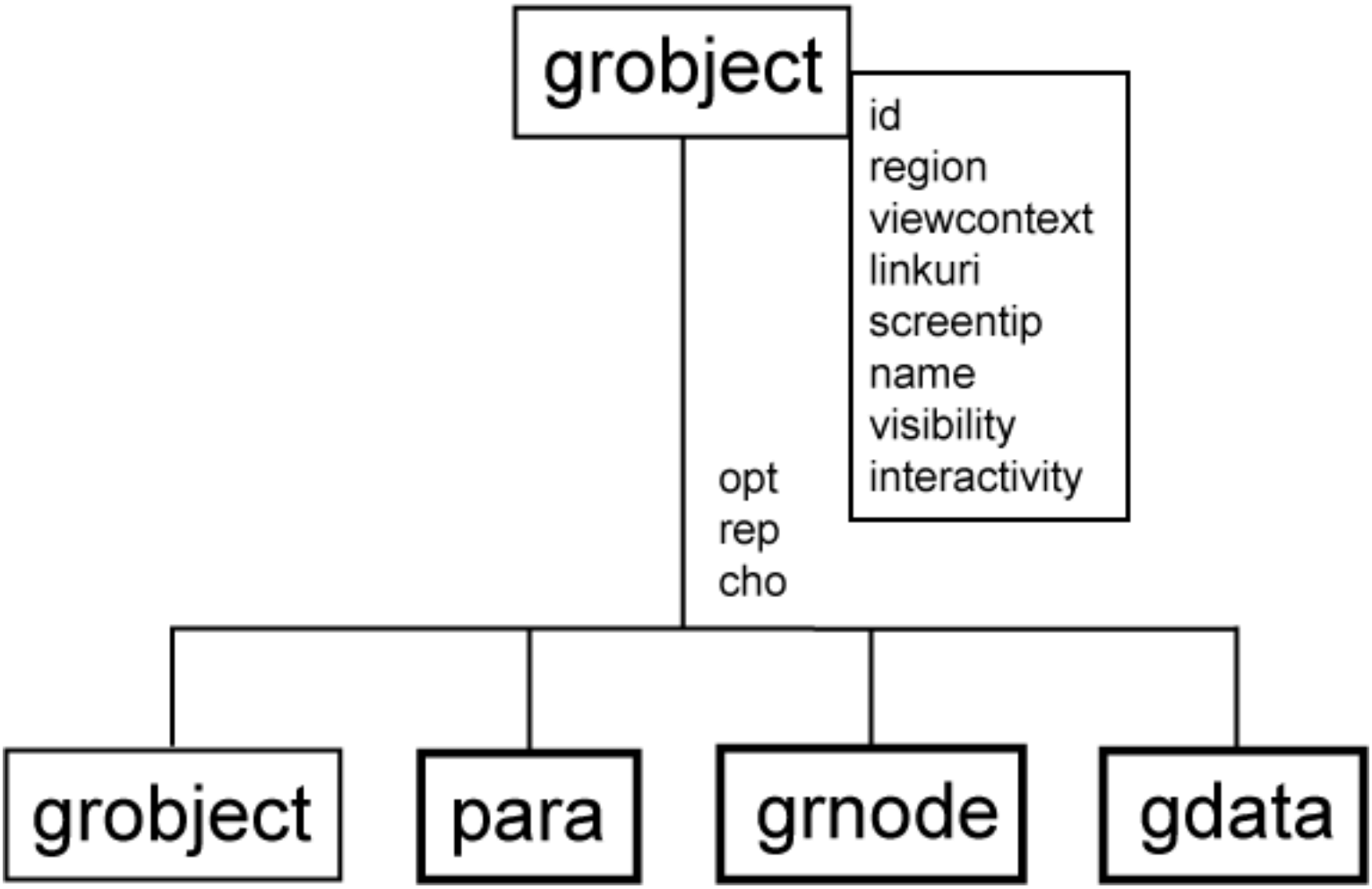

Figure 2c. WebCGM File Structure - grobject

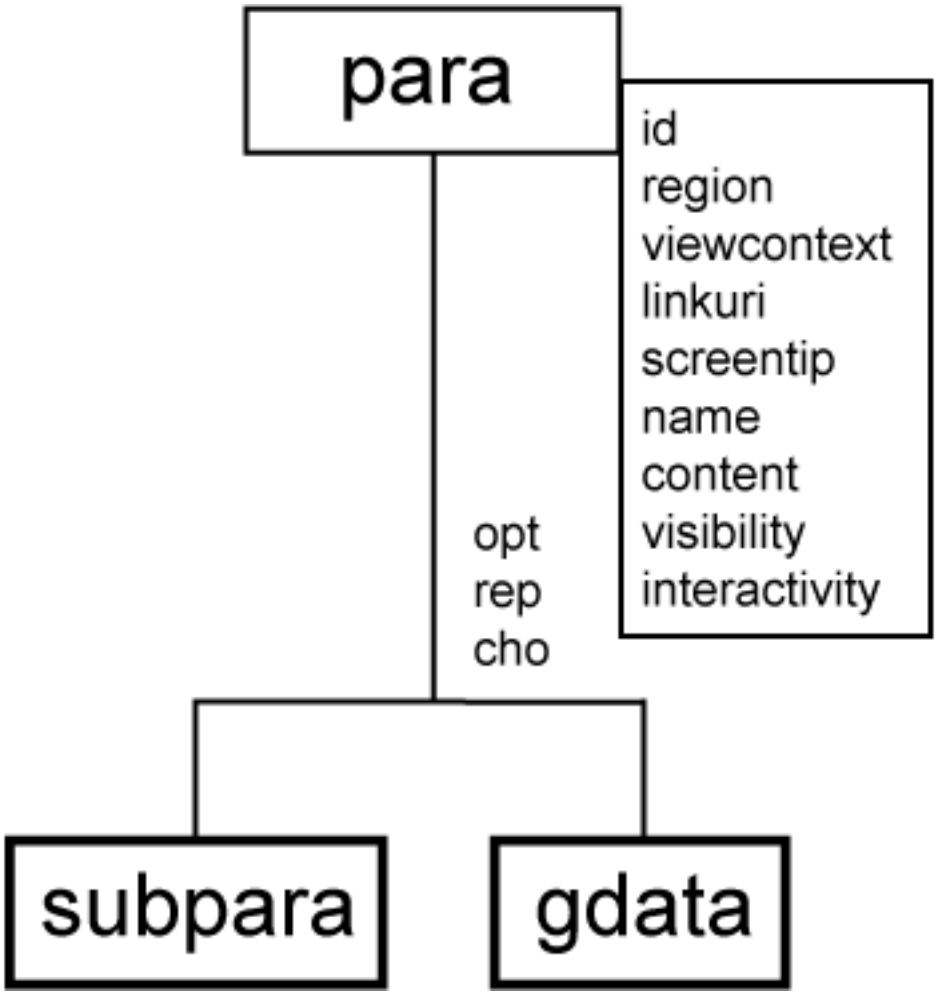

Figure 2d. WebCGM File Structure - para

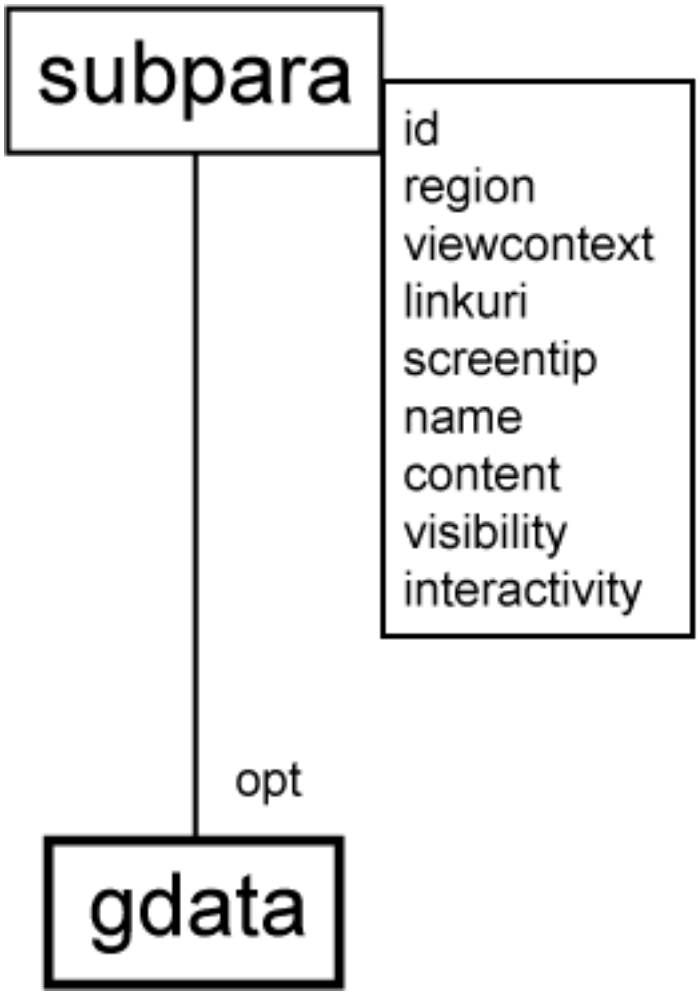

Figure 2e. WebCGM File Structure - subpara

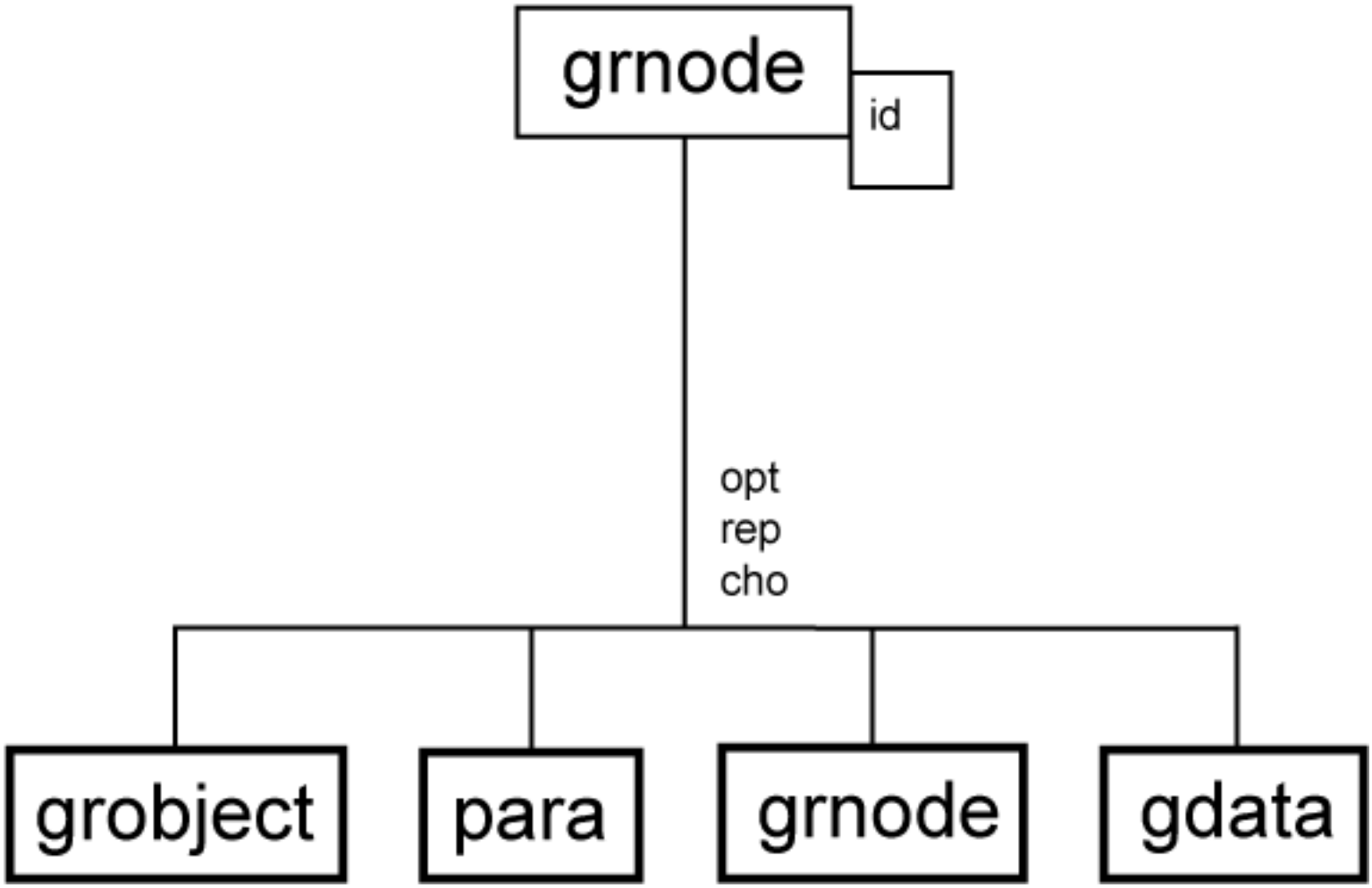

Figure 2f. WebCGM File Structure - grnode

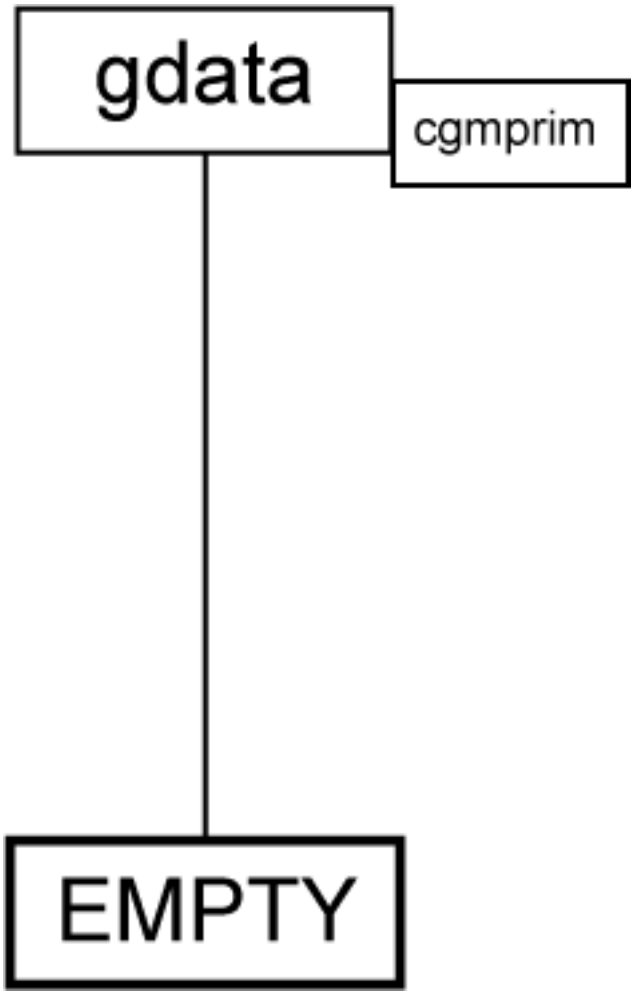

Figure 2g. WebCGM File Structure - gdata

2.3.6 Hyperlinking

WebCGM supports object-to-object hyperlinking within individual WebCGM instances, between WebCGM instances, from WebCGM instances to other Web media types, and from other media types to WebCGM instances.

In-line linking is supported, from WebCGM objects (APS of type 'grobject', 'para', and 'subpara') to WebCGM graphic files, objects, as well as to text and other media types. WebCGM fully supports linking from other media to WebCGM files and objects.

Links from WebCGM objects are realized as 'linkuri' APS Attribute elements contained within the definitions of the objects. The address of the link (a 'linkuri' parameter) is an Internationalized Resource Identifier (IRI) [RFC 3987], and is described in the normative 'linkuri' section and fragment syntax subsections.

Objects may contain multiple 'linkuri' APS attribute instances, for which case the associated Link Title

parameter is available to help the user select the destination. WebCGM prescribes a uniform viewer requirement to offer destination choice to the user for such multi-destination cases.

The target of a link, either from within a WebCGM or from another media type (e.g., HTML text), may be a WebCGM instance. WebCGM defines an optional "fragment syntax" for addressing objects within a WebCGM metafile.

The fragment syntax, in full generality, is:

<base-IRI>#<pict-part>.<obj-part>

The <pict-part> is identified by a keyword and has two pieces, the picture locator (either the 'PictureId' string parameter of the CGM, or the picture sequence number), and viewer behavior upon navigating to the picture. With the WebCGM restriction of one picture per metafile (since WebCGM version 2.0), the <pict-part> is not useful anymore, but is maintained in the syntax for backward compatibility with WebCGM 1.0 metafiles and WebCGM 1.0 implementations.

The <obj-part> similarly is identified by a keyword and has two pieces, the Id parameter of the object (APS), and viewer behavior.

The syntax is well-defined so that in many common cases, keywords and pieces can be eliminated and defaulted. So, for example, <base-IRI>#<string> unambiguously identifies the object (APS) whose Id parameter is "<string>" in the first (only) picture of the metafile pointed to by "<base-IRI>".

See the normative specifications of Chapter 3 for complete details and examples.

2.4 Encodings

ISO CGM defines two encodings of the CGM functionality: Binary, and Clear Text. WebCGM, like other leading industry profiles of CGM, limits the encoding to Binary for the purposes of conforming interchange. It is the Binary encoding which is registered as a MIME type. Using available encoding converters, the Clear Text encoding can be used for debugging, hand authoring, demonstration, etc.

HTTP/1.1 allows for compressed data to be passed from server to client, which can result in significant file size reduction. WebCGM data may be compressed for transmission using gzip compression. It is recommended that gzip-compressed WebCGM files have the extension ".cgz" (all lowercase) on all platforms.

2.5 Graphical content of WebCGM

The graphical content of WebCGM is chosen to balance the requirements of high expressive power, and simplicity to implement. It is a subset of the Model Profile (MP) of the CGM standard.

The WebCGM Profile Pro-forma (PPF), later in this document, gives the complete normative graphical content details. Following is a conceptual summary.

2.5.1 Graphical primitives

The most obvious aspect of a graphics format is the collection of graphical primitives - those drawing elements which define the geometric and other presentation content of the format. CGM:1999 contains a rich selection of vector graphics primitives, plus fully integrated state-of-the-art compressed tile raster content.

WebCGM includes most of the significant graphical drawing primitives of CGM:1999.

- The simple generic drawing primitives - polylines and disjoint polylines, polygons, polygon sets.
- A number of specialized objects, for convenience and efficiency - rectangles, circles and ellipses, circular and elliptical arcs and pie slices. These come in both unfilled (line) and filled flavors.
- Graphical Text primitives:
    - WebCGM allows the simple Restricted Text primitive of CGM:1999 (which carries its extent box with it), as well as the Append Text element (continuation of a text string, after change of attributes such as font, color, ...)
    - WebCGM also supports text-on-path: paths other than simple straight lines can be defined (similar to Compound Line), and the text is laid out along that path.
- Closed Figure and Compound Line - these primitives allow the construction of complex paths as a concatenation of any of the other line and fill primitives. The path may either be drawn according to CGM line attributes, or filled according to any of the permissible fill attributes.
- Smooth curves, available in two flavors:
    - the basic and popular piece-wise cubic Bezier capability of the CGM:1999 Polybezier element;
    - more powerful Non-uniform B-splines (NUBS) and Non-uniform Rational B-splines (NURBS).
- Raster capabilities are available in two flavors, fully integrated with the scalable vector functionality of WebCGM:
    - the simple, uncompressed (except for run-length) Cell Array element of CGM:1999;
    - and the compressed, tiled Tile Array capability of CGM version 3. The popular Web compression formats of ITU-T Group 4, PNG, and JPEG are among the supported compression types.

In CGM:1999 but excluded from the present version of WebCGM are:

- The unrestricted TEXT element is prohibited (the results are unpredictable, so the more reliable RESTRICTED TEXT is required).
- The Hyperbolic Arc and Parabolic Arc elements of CGM:1999 are prohibited (these are seen in some advanced engineering formats, but not yet in Web practice).
- Polysymbol - the Polysymbol element allows the sizing and placement into CGM pictures of "symbols". Symbols are defined in an external Symbol Library, which itself is a CGM. (Polysymbol was in WebCGM 1.0, but removed due to non-use.)

2.5.2 Attributes and controls

Attribute elements and control elements determine the details of the appearance of graphical primitives.

- Line attributes of WebCGM include line dash styles, line width, line color, and the controls over appearance of line cap and line join. Dash styles can either be predefined, or can be defined precisely by the generator of the metafile.
- Fill attributes control the appearance of the interior of filled primitives. One attribute controls the overall style of the filled-area interior - solid color, hatch, bitmap-style pattern, empty, and hollow. In the case of hatch interior, the style can either be one of a handful of predefined, or can be precisely defined in the metafile. The pattern can be defined as a small "raster tile" of two or more colors.
- The related Edge attributes control the appearance of edges of filled areas, and are for the most part identical to Line attributes.
- The appearance of graphical text can be controlled by the CGM attributes of font and character set (which corresponds to *character encoding* in the proper terminology of *Character Model for the World Wide Web 1.0: Fundamentals* [CHARMOD], see later), text size, orientation, inter-character spacing, expansion-compression of nominal character aspect ratios - the important aspects needed for precise control in modern graphical text presentation. The orientation attribute actually gives control not just of rotation, but skewness and aspect distortion, i.e., it is equivalent to a local transformation on text elements.
- WebCGM also supports the Generalized Text Path capability (text on arbitrary, curved paths), with the style (Generalized Text Path Mode) limited to 'axis-tangential'.
- WebCGM clipping support includes:
    - rectangular regions;
    - arbitrary complex clip regions (Protection Region, similar to Closed Figure), with Protection Region Indicator restricted to 'clip'.

The following attribute and control features of CGM:1999 are excluded from WebCGM.

- The use of the bundled attributes model of CGM Version 1 is prohibited, which eliminates all of the Version 1 bundle index elements, the ASF elements, and the bundle-table setting elements of Version 2.
- The use of the CGM Version 2 Segment functionality is prohibited, and its dozen or so associated elements may not occur in valid WebCGM metafiles.
- The Geometric Pattern fill capability is prohibited.
- Shielding functionality is excluded, i.e., Protection Region with mode (Protection Region Indicator) 'shield'.

There are some CGM Version 3 attribute and control elements for which it is desirable to override the default value in CGM:1999, when an explicit definition of the value is not present in the CGM file. This would also allow definition of the rendering behavior of CGM Version 1 and Version 2 files, where those attribute and control elements are not allowed, as well as allow definition in CGM Verson 3 files where the elements are not declared.

This is accomplished in WebCGM using a standard XML DTD to encode the allowable elements and their values in an XML instance. Examples and more details can be found in the WebCGM chapter on

Application Configurable Items.

2.5.3 Color and transparency

The normal behavior of CGM:1999 viewers is to render later occurring primitives completely opaquely on top of earlier primitives. Several notions of transparency are supported in WebCGM. See section 2.2.2 and 2.2.3 for discussion of the CGM drawing model and transparency options.

The full range of standard CGM:1999 color models is limited in WebCGM. The default RGB model is included, as well as the models: RGB-alpha; the colorimetric RGB space of the Web, sRGB; and sRGB-alpha. The latter three are registered in the ISO Register of Graphical Items.

2.5.4 Character encodings and fonts

Fully international text is supported in WebCGM by:

- Allowing the Unicode UTF-8 or UTF-16 character encodings to be selected in the CGM's character encoding designation and selection mechanisms (CHARACTER SET LIST, CHARACTER SET INDEX, etc).
- Requiring the use of the CGM:1999 Font Properties element, in the case of fonts outside of a required core set of 13. The Font Properties element carries a handful of parameters which are descriptive of a set of font attributes which are most commonly used to classify and call out fonts.

The default character encoding ("character set" in the now-archaic terminology of the original CGM:1987) is *ISOLatin1*. This default is mandated by the ISO CGM:1999 standard.

A core set of 13 fonts, the same as those in the ISO CGM Model Profile (MP), are required in WebCGM implementations.

In order to facilitate font interchange, WebCGM defines a format to specify the mapping of font names during the import process.

This mapping is accomplished in WebCGM using standard XML DTD. Examples and more details can be found in the WebCGM chapter on Application Configurable Items.

2.6 WebCGM XML Companion File (XCF)

The XML Companion File (XCF) component of WebCGM was added in the WebCGM 2.0 release. The WebCGM XCF provides a standard way to externalize metadata from a WebCGM instance, while maintaining a tight binding of that metadata to objects (APSs) in the WebCGM instance.

The WebCGM XCF was designed with three main usage scenarios in mind. A WebCGM companion file:

1. can be used to bind application specific metadata (such as a part number) to a particular Application Structure in a WebCGM illustration.
2. could also be used to update metadata in a WebCGM illustration via the WebCGM DOM (see DOM section *Relationship with XML companion file* for more information).
3. could be used as a partial inventory of a WebCGM illustration by enumerating the Application Structures IDs, types and (most) attributes. (Note that it is out-of-scope of this version of WebCGM XCF to fully mirror the hierarchical structure of a CGM graphic (see "Structure overview" in the XCF chapter.)

Examples and more details may be found in the *WebCGM XML Companion File* (XCF) chapter, the complete normative definition of the WebCGM XCF.

The normative definition of XCF includes a base and generic DTD. The WebCGM XCF is designed to be extensible, by other profiles derived from WebCGM, as well as applications of WebCGM. In particular, this allows industry specific metadata to be added to the WebCGM object model. See the normative XCF definition for details.

The XCF is a mechanism to bind external metadata to objects in WebCGM instances. Accordingly, unlike hierarchical tree structured WebCGM instances, the structure of the XML Companion File is mostly flat. See the normative section, *Relationship with XML companion file*, for more details.

2.7 WebCGM Document Object Model (DOM)

2.7.1 Motivation

The Document Object Model (DOM) component of WebCGM was added in the WebCGM 2.0 release. An interface for programmatic access to WebCGM contents and structure, as well as facilities to manipulate a standardized WebCGM XML Companion File, were perhaps the strongest driving requirements for the WebCGM 2.0 release. Virtually all of the WebCGM viewer and user agent implementations had already defined and implemented a proprietary application programming interface (API) for such functionality.

2.7.2 Scope of WebCGM DOM

Compared with detailed, complete DOM specifications such as W3C's XML DOM Level 3, or the DOM of the SVG 1.1 Recommendation, the WebCGM DOM has limited scope. A full DOM would support query and discovery of all objects and entities in a target content (graphic) instance, right down to the leaf nodes of the structure tree. It would also support symmetric, detailed modification and manipulation capabilities for changing the object.

The functionality available in the WebCGM DOM is somewhat more limited. The WebCGM DOM exposes the document graphic structure down to the Application Structure (APS) level -- APS's are the fundamental addressable graphical objects in WebCGM, and are the building blocks of the hierarchical

structure tree of a WebCGM.

The WebCGM DOM supports transient manipulation of the APS attributes -- which represent non-graphical metadata associated with the objects -- and transient changes to presentation style of graphical objects. The WebCGM DOM provides functionality to support query and discovery of the structure of a WebCGM, enumeration of its graphical objects, extraction of associated metadata (e.g., hyperlinking data) from documents, and finally provides users with *standard* ways to add more interactivity to WebCGM documents than was previously possible.

The WebCGM DOM also provides functionality for manipulation and application of standard WebCGM XML Companion Files, described in the previous section.

The WebCGM DOM supports a number of usage scenarios and gives access to a number of useful capabilities. Collectively, the *WebCGM 2.0 Requirements* and the *WebCGM 2.1 Requirements* documents give details about the in-scope and out-of-scope capabilities of WebCGM DOM.

Back to top of chapter

---

3. WebCGM Intelligent Content

This chapter and its sections are normative, unless otherwise indicated.

Contents

3.1 Addressing objects

3.1.1 IRI fragment specification

3.1.1.1 Fragment definition

The IRI (Internationalized Resource Identifier) is how resources are identified on the Web. For example, a CGM file called web.cgm might have the following IRI:

```
http://example.org/web.cgm
```

Application structures and pictures within a WebCGM are addressed using the mechanism of the IRI fragment identifier. These WebCGM rules are derived from and are consistent with the Web protocols defined in IRI [RFC 3987] and URI (Uniform Resource Identifier) [RFC 3986].

An IRI which includes an IRI fragment identifier consists of an optional base IRI, followed by the separator character "#", followed by the IRI fragment identifier. For example, the following IRI can be used to specify the "wheelAssembly" object within web.cgm:

```
http://example.org/web.cgm#wheelAssembly
```

The fragment identifier is usually specific only to a particular class of applications. This clause defines the WebCGM fragment identifier which allows WebCGM viewers, web browsers, scripting engines, and other applications to:

- address (point to) specific objects within WebCGM files,
- or load and apply an XML Companion File (XCF).

The IRI fragment syntax, as adopted by WebCGM, is based on concepts described in the *XML Pointer Language* (see *XPointer Framework*). The IRI fragment syntax is defined below. The formal grammar for the WebCGM fragment is given using a simple Extended Backus-Naur Form (EBNF) notation.

Note 1. Multiple pictures per WebCGM instance were allowed in WebCGM 1.0. Since 2.0, WebCGM allows only one picture per metafile. For backward compatibility with existing 1.0 metafiles and with viewer implementations, the fragment syntax is unchanged with regard to the picterm.

Note 2. WebCGM 1.0 described links using URI terminology, however implementations interpreted the WebCGM 1.0 language differently, and in fact some essentially supported IRIs (Internationalized Resource Identifiers). IRIs are a more generalized complement to Uniform Resource Identifiers (URIs). An IRI is a sequence of characters from the Universal Character Set [Unicode40]. A URI is constructed from a much more restricted set of characters. All URIs are conformant IRIs. A mapping from IRIs to URIs is defined by the IRI specification, and this mapping is adapted to WebCGM below (3.1.1.4). IRIs can be converted to URIs if the WebCGM processor does not support IRIs directly. In this specification, the correct IRI terminology is used.

3.1.1.2 Fragment EBNF

```
The following notation is used:
```

```
    *  – 0 or more
    +  – 1 or more
    ?  – 0 or 1
    ;  – separates multiple fragment entries
    () – grouping
    |  – separates alternatives
    double quotes (") surround literals


webcgmfragment ::= picterm "." objterm |
            picterm |
            objterm  |
            picid "." objid |
            objid |
            xcfterm

picterm ::= pictureid | pictsequence

pictureid ::= "pictid("   picid   ("," behavior)? ")"

picid ::= (char)+

behavior ::=  "_blank" | "_self" | "_parent" | "_replace" | "_top" | target

target ::= (char)+

pictsequence ::= "pictseqno(" picseqno ("," behavior)? ")"

picseqno ::= (digit)+

digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

objterm ::= specialForm | normalForm

specialForm ::= "id(*,clearHighlight)"

normalForm ::= objectid | objectname

objectid ::= "id("   objid   ("," objbehavior)? ")"

objid ::= (char)+

objBehavior ::= navTerm | highlightTerm | navTerm "+" highlightTerm

navTerm ::= "full" | "zoom" | "move"

highlightTerm ::= "newHighlight" | "addHighlight"
```

```
objectname ::= "name("   objname  ("," objbehavior)? ")"

objname ::= (char)+

xcfterm ::= "xcf(" xcfurl ")"

xcfurl ::= (char)+
```

See next section for a definition of the "char" production.

3.1.1.3 Fragment Character Repertoire

The productions '`picid`', '`target`', '`objid`', '`objname`', and '`xcfurl`' in the fragment grammar above are represented by parameters in WebCGM content of type non-graphical text (CGM type SF). Their character repertoire shall be restricted as follows.

1. Firstly, per the character set for type SF data. See section 6.3, T.14.5.
2. Secondly, the character repertoire for all of these productions is further restricted as defined in section 2.2 of XML 1.0, fourth edition.
3. Thirdly, the repertoire for each of these productions is further restricted as follows:
    - `objid` - corresponds to WebCGM Begin APS 'id' parameter; further restrictions: as the *name* construct defined in section 2.3 of XML 1.0, fourth edition.
    - `objname` - corresponds to WebCGM APS attribute of type 'name'; shall not contain (leading, trailing, or embedded) any of the whitespace characters #x09, #x0a, or #x0d, and shall not contain any leading or trailing blanks (#x20); the 1-character string "*" is reserved for special meaning and is not a valid `objname`; no further restrictions.
    - `picid` - corresponds to the 'id' parameter of WebCGM BEGIN PICTURE elements; further restrictions: as objid for any picid value occurrences within a fragment, otherwise per type SF for the id parameter in the BEGIN PICTURE element itself (see further comments below).
    - `target` - further restrictions: as `objid`, plus must begin with [a-zA-Z], similarly to "Frame Target Name" of the *HTML 4.01 Specification*, upon which the set of WebCGM picture behaviors is based.
    - `xcfurl` - must be a valid IRI, according to the encoding rules below (3.1.1.4), and follow the rules specified for the first parameter of the 'linkuri' Application Structure Attribute.

Note that these character repertoires allow one or more of the characters ".", ",", "(", and ")". These are significant characters in the syntax of the WebCGM fragment specification. If any of these four significant characters is to appear in a valid id/name string within a fragment instance, then the fragment shall use the unabbreviated long form, which is the first of the six optional forms in the 'webcgmfragment' production of 3.1.1.2. In particular, all components of the long form shall be included, and none of the parts marked as optional in the EBNF may be omitted.

Note about `picid`: The character repertoire for `picid` occurrence in the fragment ("as `objid`") is more restrictive than the repertoire for the id parameter of the BEGIN PICTURE element itself (CGM data type SF, not further restricted). Any application which intends to use the `picid` field in the fragment must generate the picture ids to the more restrictive repertoire of the fragment. (See further discussion of picture selection keywords in fragments, section 3.1.2.1.)

3.1.1.4 Non-URI characters in IRIs

The URI character repertoire, as defined in RFC 3986, is comprised of the alphabetic and numeric characters of ASCII, plus a few punctuation marks. The character repertoires defined in 3.1.1.3, and allowed in IRIs, are much richer. The method for handling this disparity is described in this section.

An IRI in WebCGM content must be a URI reference as defined in RFC 3986, or must result in a URI reference after the 3-step escaping procedure is applied as described in RFC 3987 (IRI), section 3.1 ("Mapping of IRIs to URIs"), Step 2. The procedure is applied when a WebCGM processor (viewer, or any other WebCGM-interpreting process) passes the URI reference to a URI resolver.

This handling allows various degrees of " URI legal" (in the sense of RFC 3986-conforming) content to appear in WebCGM content, as shown in the examples at the end of this section. In the WebCGM context, WebCGM processors can practically determine when to apply the 3-step escaping procedure as follows: if an IRI in WebCGM content contains a character sequence that corresponds to a valid URI escaping sequence, i.e., a three-character "%HH" sequence, WebCGM processors shall consider that sequence to be a URI escaping sequence. Such a sequence may be passed directly to the URI resolver. (This is shown in the examples.)

This does still not guarantee that a URI will result that is valid according all applicable URI specifications. E.g., it could result in DNS-illegal characters in the domain name, if the input WebCGM content was badly formed its generator. Because it is impractical for any application to check that a value corresponds to a valid URI, this specification follows the lead of RFC 3986 in this matter and imposes no such conformance testing requirement on WebCGM applications. Although no URI conformance testing requirements or error handling are specified here, WebCGM processors should, of course, react gracefully to bad input.

EXAMPLES:

string in WebCGM	disposition and handling
`my WebCGM.cgm`	must be URI-escaped, `my%20WebCGM.cgm`, before passing to URI resolver.
`my%20WebCGM.cgm`	is already URI-escaped (represents "`my WebCGM.cgm`"), may be passed directly to URI resolver.
`%clear text comments%`	not URI-escaped, must be converted to `%25clear%20text%20comments%25` before passing to URI resolver.

%25123456%25	already URI-escaped (represents "%123456%"), may be passed directly.
%25123456%	partially URI-escaped (represents %123456%), must be fully escaped (%25123456%25) before passing.
2 Japanese chars: u+65e5, u+672c	must be URI-escaped — first converted to UTF-8: EF BB BF E6 97 A5 E6 9C AC (9 octets), then non-ASCII chars are %-escaped: %EF%BB%BF%E6%97%A5%E6%9C%AC

3.1.1.5 Resolving relative IRIs for XCFs

The above fragment EBNF defines a `xcfterm` production whose '`xcfurl`' parameter identifies an XML Companion File associated the WebCGM identified by the IRI. WebCGM viewers that support XCF must interpret the fragment, locate and load the XCF. The '`xcfurl`' parameter is itself a IRI, and it can be absolute or relative. WebCGM viewers shall resolve a relative '`xcfurl`' IRI relative to the IRI of the WebCGM instance with which the resource file is a companion — i.e. relative to the WebCGM file referenced by the base part of the IRI containing the fragment — rather than relative to the file containing the IRI reference (e.g., an HTML file).

EXAMPLE. Suppose an HTML document contains a hyperlink (`href` attribute on <a> element) to a WebCGM illustration, and that hyperlink IRI contains a fragment specifying to load and apply an XML Companion File:

- HTML at: `http://www.example.org/parts-list.html`
- contains href: `http://www.example.org/illustrations/some-part.cgm#xcf(some-part.xml)`

Then the relative IRI "`some-part.xml`" is resolved according to the location of the associated CGM, not according to the location of the HTML:

- `http://www.example.org/illustrations/some-part.xml`

Similarly, if the fragment were `#xcf(companions/some-part.xml)`, then that IRI resolves to:

- `http://www.example.org/illustrations/companions/some-part.xml`

3.1.2 Fragment parameters

The following subsections describe in detail how fragment parameters are used to specify the picture behaviors and object behaviors of links containing the parameters. An informative summary of the rules is provided in the last subsection of this section (3.1.2.8).

3.1.2.1 Picture selection keywords

Since version 2.0, WebCGM allows only one picture per metafile. Therefore the picture selection keywords are of limited utility, However, WebCGM 1.0 allowed multiple pictures per metafile. Newer WebCGM viewers might therefore operate in legacy 1.0 environments or "mixed" environments:

- WebCGM 2.1 & 2.0 viewers, especially when they are upgrading and replacing WebCGM 1.0 viewers, might receive external requests to link to multi-picture WebCGM 1.0 metafiles;
- WebCGM 2.1 & 2.0 metafiles might need to link to legacy WebCGM metafiles that are multi-picture;
- WebCGM 2.1 & 2.0 viewers might receive requests for non-existent pictures in metafiles of any version, and there should be fallback behavior as there was in WebCGM 1.0.

`pictid` - The `pictid` keyword indicates that the picture to be viewed is identified by the id of the picture, which is the id parameter in the BEGIN PICTURE element. In the syntax, this is a required parameter of the `picterm` production, and there may be a second associated parameter, whose value is an optional picture behavior specification (see EBNF). If the metafile does not contain a picture with the specified picture id value, the first (and only) picture in the metafile is chosen.

`pictseqno` - The `pictseqno` keyword indicates that the identity of the picture to be viewed is by the sequence number of the picture in the metafile (see EBNF). In the syntax, this is a required parameter of the `picterm` production, and there may be a second associated parameter, whose value is an optional picture behavior specification. The picture sequence number shall be an integer greater than 0. If the specified picture sequence value exceeds the number of pictures in the metafile, the last picture is displayed.

3.1.2.2 Picture behaviors

Picture behaviors describe to the viewer how to display the remote resource of a hyperlink. Picture behaviors are based on the syntax and semantics of Frame Target Names defined in the *HTML 4.01 Specification*. The "picture behaviors" concept of WebCGM is used to collectively address CGM pictures and other Web resources. "Picture behaviors" is use to specify the name of a relevant presentation context (e.g., an HTML or XHTML frame, iframe, or object element) into which a document is to be opened when the link is activated

The reserved names listed below describe the various picture behaviors. All other Picture Behavior values shall follow the naming convention of the presentation context in question. For example, if the presentation context is an HTML frame, then the naming convention for Frame Target Names in the *HTML 4.01 Specification* is followed — if not one of the reserved keywords, they must begin with [a-zA-Z]."

In what follows, the following conventions apply:

- "viewer" refers collectively to a browser and/or a CGM viewer (e.g., plugin). The logical result is described, not the method of achieving the result (which in general will require action by browser or viewer or both.)
- "content" refers to either source or destination of a link, and is either a picture (CGM) or another Web resource, depending upon the formulation of the particular link.

The following Picture Behavior values, except for _replace, are based on Frame Target Names of HTML 4.01:

_blank
: The viewer shall load the designated content in a new, unnamed window.

_self
: The viewer shall load the content in the same frame as the one containing the content that refers to this target.

_parent
: The viewer shall load the content into the immediate FRAMESET parent of the current frame in which the current content is displayed. This value is equivalent to "_self" if the current frame has no parent.

_replace
: When _replace occurs on a CGM-to-CGM link, a WebCGM viewer shall replace the current CGM picture by the designated CGM picture in the same rectangular area in the same frame as the picture which refers to this target. If the ending resource (CGM) is the same as the linking resource, the viewer shall not reload the resource. This is the default behavior for such links..

_top
: The viewer shall load the content into the full, original window (thus canceling all other frames). This value is equivalent to _self if the current frame has no parent.

If the picture behavior value is the name of a relevant presentation context (e.g., an HTML or XHTML frame, iframe, or object element), the remote resource shall be displayed in the specified presentation context. If the presentation context already exists, it is re-used, replacing the existing content. If it does not exist, the viewer shall load the designated content (picture or document) in a new window with the specified name. Note that frame-target must be an XML Name [XML10].

WebCGM objects are often embedded in a parent language. In such a situation, link activation behaviors (picture behaviors) are defined by the parent object's relevant language. For example, if the parent language were HTML, then the link activation behavior would be defined by the *HTML 4.01 Specification* [HTML401].

> **EXAMPLE**. If the parent language is HTML (per the *HTML 4.01 Specification*) and the referenced resource is CGM, the links are **HTML-to-CGM** links. The effect of picture behaviors should be achieved with an HTML '`target`' attribute on an <a> element, for example:

```
<a href="someCGM.cgm" target="_blank" ... />
```

The effect should *not* be achieved by picture behavior terms in the fragment syntax. I.e., the following is invalid usage and viewers should ignore such fragment specifications:

```
<a href="someCGM.cgm#picseqno(1,_blank)" ... />
```

EXAMPLES.

The following table illustrates how the generalized rules for link activation behavior (picture behaviors) would correctly be applied for some common source and destination resource types (HTML and CGM), and for the specific HTML presentation context of frames. (See caveat following the table.)

Picture behaviors and different source-to-destination data types.

Behavior	CGM-to-HTML	CGM-to-CGM
_blank	The viewer shall load the designated document in a new, unnamed window.	The viewer shall load the designated picture in a new, unnamed window.
_self	The viewer shall load the document in the same frame as the one containing the CGM picture that refers to this target. This is the default for CGM-to-HTML links.	The viewer shall load the picture in the same frame as the one containing the CGM picture that refers to this target.
_parent	The viewer shall load the document into the immediate FRAMESET parent of the current frame in which the current picture is displayed. This value is equivalent to _self if the current frame has no parent.	The viewer shall load the picture into the immediate FRAMESET parent of the current frame in which the current picture is displayed. This value is equivalent to "_self" if the current frame has no parent.
_replace	Not applicable. Default to _self.	The viewer shall display the designated CGM picture in the same rectangular area in the same frame as the picture which refers to this target. If the ending resource (CGM) is the same as the linking resource, the viewer does not reload the resource. This is the default behavior for CGM-to-CGM links.

_top	The viewer shall load the document into the full, original window (thus canceling all other frames). This value is equivalent to _self if the current frame has no parent.	The viewer shall load the picture into the full, original window (thus canceling all other frames). This value is equivalent to _self if the current frame has no parent.
target	The viewer shall load the document into the frame identified by "target". If no matching frame can be found, the viewer shall load the designated content document in a new window with the specified name.	The viewer shall load the picture into the frame identified by "target". If no matching frame can be found, the viewer shall load the designated content document in a new window with the specified name.

Note: This table comprised the entire normative rules of WebCGM 1.0 for "picture behaviors". However, it did not include sufficient generality in the possible resource types, nor in different presentation contexts. Therefore it is retained as examples of correct application of the current, more generalized rules of WebCGM.

Figures 3 and 4 below give examples of _self and _replace.

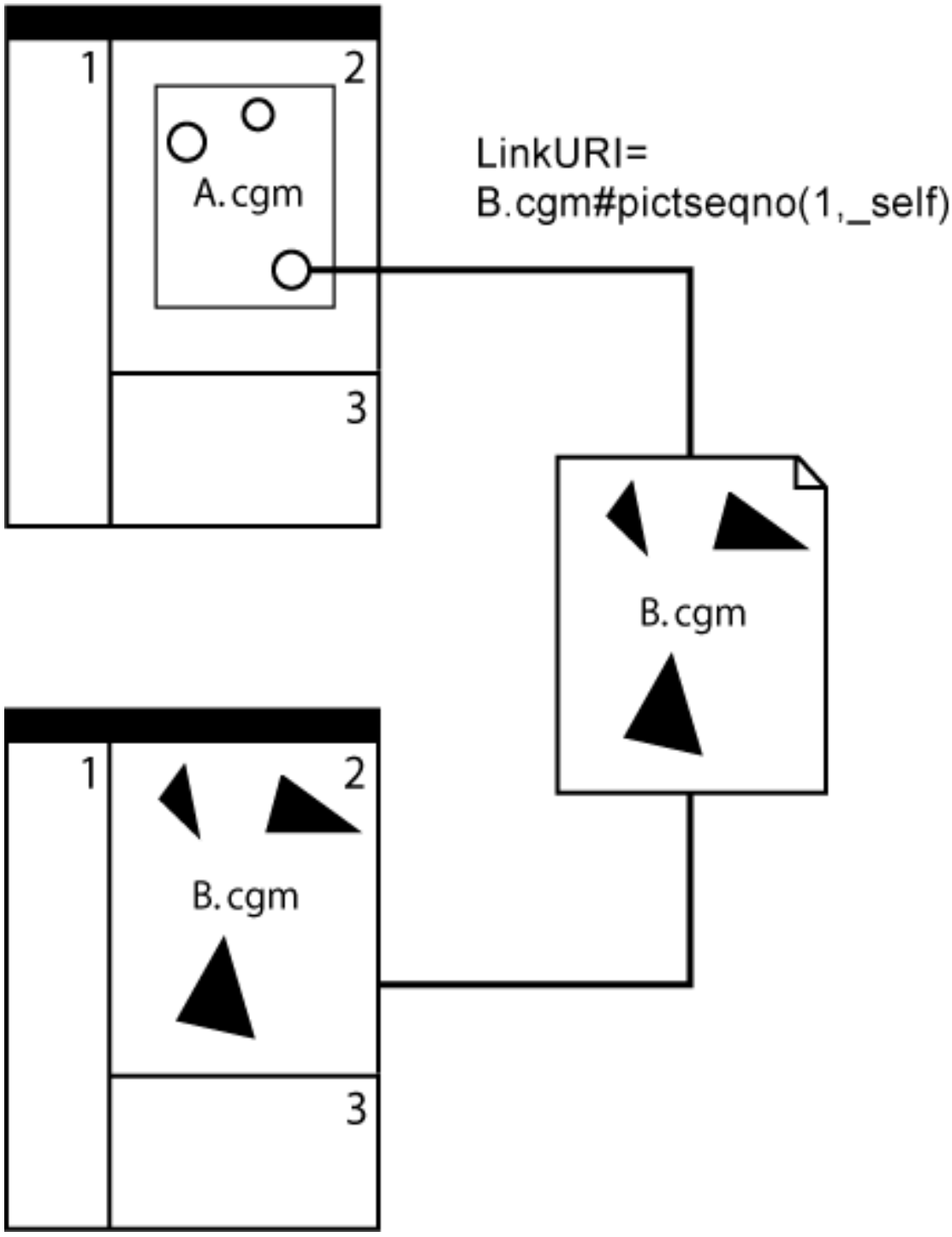

Figure 3. Example of _self

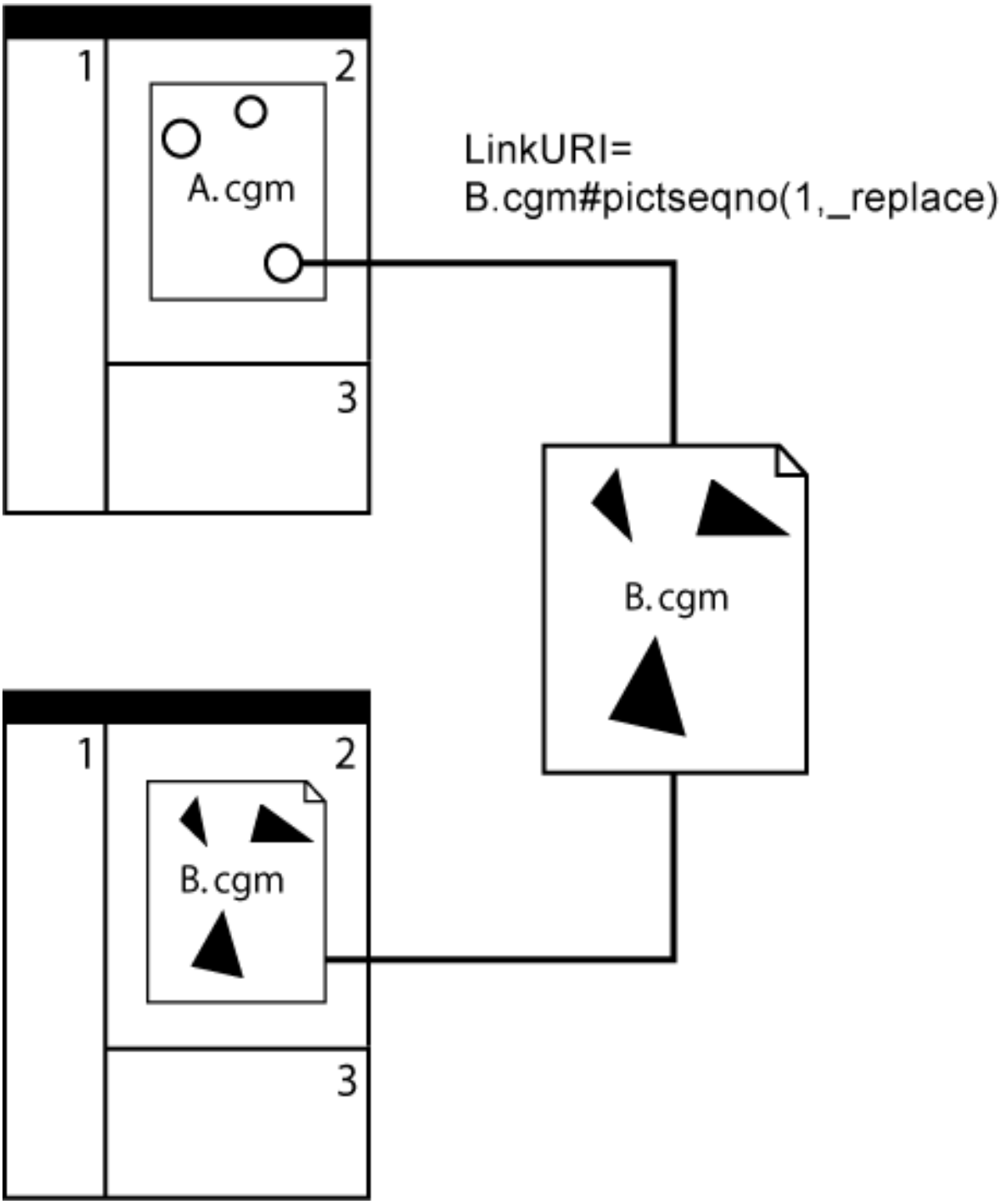

Figure 4. Example of _replace

3.1.2.3 Object selection keywords

`id` - the id of the APS of type 'grobject', 'para' or 'subpara' to be selected. The id parameter in the BEGIN APS element. If no match is found in the picture, no object is selected.

`name` - the value of the 'name' attribute in an object (grobject, para, or subpara APS). This is an alternate way to address objects. All objects in the picture with matching 'name' attributes are selected. If no match

is found in the picture, no object is selected.

In addition to these two object selection keywords, WebCGM defines one special form object behavior that allows a wild-card object selection keyword, `"*"`. The wild-card keyword shall be used only in this special form.

3.1.2.4 Object behaviors

Object behaviors describe how to present the view of object(s) that is (are) the targeted by a hyperlink containing a IRI fragment that selects a particular object or group of objects as the target.

3.1.2.4.1 Enumeration of behaviors

WebCGM contains twelve distinct object behaviors, eleven of which are generated by the fragment EBNF, and one of which is defined in the EBNF as a **special-form object behavior** fragment:

`id(*,clearHighlight)`

The default object behavior is `zoom+newHighlight.`

The next subsection defines the target rectangle of the navigation behaviors, depending on whether a single or multiple objects are selected by the object selection keyword. The highlight behaviors apply to all objects selected according to the object selection keyword.

The object behaviors are built from a set three "atomic" navigation keywords and a set of two "atomic" highlight keywords, defined below. The navigation keywords have no highlighting side effects and the highlighting keywords have no navigation side effects — the navigation and highlighting keyword sets are orthogonal.

The special-form behavior, `id(*,clearHighlight)`, clears highlighting of any and all highlighted objects in the picture, without affecting the navigation state of the picture.

Note that object behaviors are "cumulative". As described in the "Picture behaviors" section, if a link points to the same CGM file that is currently viewed — e.g., a CGM-to-CGM link within the same file, with `_replace` picture behavior — then the picture is not reloaded, and any new object behaviors are applied starting from the current viewing state of the picture.

The following table enumerates the object behaviors available in WebCGM. The *Navigation* column indicates whether the viewer performs any navigation for the behavior, for the selected object(s). *Highlighting* indicates whether the viewer adjusts the highlighting of the selected object(s) in some way ("X") or not.

Object behaviors of WebCGM

Behavior	Navigation	Highlighting
full	X	
zoom	X	
move	X	
newHighlight		X
addHighlight		X
full+newHighlight	X	X
zoom+newHighlight (default)	X	X
move+newHighlight	X	X
full+addHighlight	X	X
zoom+addHighlight	X	X
move+addHighlight	X	X
clearHighlight		X

WebCGM 1.0 defined three object behaviors that WebCGM 2.0 deprecated — view_context, highlight, and highlight_all. WebCGM 2.1 viewers shall support these three object behaviors, and shall do so by mapping them to WebCGM 2.1 behaviors as follows:

- view_context maps to zoom+newHighlight
- highlight maps to full+newHighlight
- highlight_all maps to full+newHighlight

3.1.2.4.2 Definition of target rectangle

If navigation to an object or group of objects is indicated, a target rectangle must be calculated as follows:

- If the object APS contains a 'viewcontext' APS Attribute, the target rectangle is described by the 'viewcontext' APS Attribute.
- If the object APS does not contain a 'viewcontext' attribute, but contains a 'region' attribute, the target rectangle is the rectangle enclosing the region defined by the 'region' APS attribute.
- If the object APS contains neither a 'viewcontext' nor a 'region' attribute, the target rectangle is the bounding box of the graphical primitives of the object.
- For multiple selected objects the target rectangle is the bounding box of the target rectangles of the individual objects, which are computed according to the preceding 3 bullets.

3.1.2.4.3 Definition of behaviors

full

The viewer shall present a full picture view of the appropriate picture.

zoom

The viewer shall fit the target rectangle of the selected object(s) into the viewer's rectangle and center it.

move

The viewer shall move (pan) the illustration such that the target rectangle is centered inside the viewer's rectangle. If this would cause the VDC Extent boundary to move to the interior of the viewer's rectangle, it is implementation dependent whether or not the viewer performs that part of the centering operation. If the target rectangle is too large, the viewer shall fit the target rectangle into the viewer's rectangle and center it.

newHighlight

Highlight the selected object(s), first removing any highlighting that may exist on objects in the picture.

addHighlight

Highlight the selected object(s), leaving highlighted any objects in the picture that may be already highlighted.

clearHighlight

See the definition of the **special-form object behavior** fragment in the enumeration of behaviors.

3.1.2.5 Zoom and pan

In addition to being able to meet the zooming and panning requirements imposed by the above object behavior specifications, and those zooming and panning requirements imposed by support of the `object` element's parameters, viewers which operate in an interaction-capable environment shall have zoom and pan controls available to the user. The exact methods and user interface styles for zoom and pan selection and manipulation are viewer dependent.

3.1.2.6 XML Companion File

The fragment syntax can be used to load and apply an XML Companion File (XCF) together with the WebCGM file. An interpreter that supports WebCGM DOM shall load the WebCGM file first. Then it shall load and apply the XML Companion File specified by the IRI (resolving a relative IRI if necessary). Finally it shall apply the XCF before the first display of the CGM.

Interpreters that do not support the WebCGM DOM may ignore this fragment type.

3.1.2.7 Summary of behaviors

The following is *informative* summary and is intended to provide references to where *the normative* picture and object behaviors are defined.

- For links that contain an IRI with an IRI fragment and have a CGM object as the remote resource, object behaviors are specified as described in section 3.1.2.4.

- For link activation within a parent object (e.g., HTML) the behavior is defined by the parent object's relevant language specification (ex: HTML '`target`' attribute of the <a> element) (see 3.1.2.2). Any picture behavior in an IRI fragment will be ignored by viewers (see 3.1.2.2).
- For links originating from a CGM resource, the picture behavior is to be expressed using the third parameter of the `linkuri` Application Structure Attribute (see 3.2.2.3). Note that encoding the picture behavior in the first parameter of the `linkuri` Application Structure Attribute is discouraged when targeting a CGM resource (see 3.2.2.3, "discouraged").
- For **DOM script to CGM links** the `src` attribute of the `WebCGMMetafile` interface specifies the IRI of the image. Specifying behavior as part of the IRI is prohibited (see 5.7.3 'src').

3.1.3 Examples

This subsection and its subsections are informative (non-normative).

3.1.3.1 Preliminaries

The WebCGM fragment in its most verbose form provides the means to address objects between metafiles, and tells the viewer what to do to execute the link. The default viewer behavior defines what the browser shall do if the WebCGM fragment does not explicitly define the viewer behavior.

The following examples illustrate some of the ways the WebCGM fragment can be used. The examples describe how one might address a set of CGM files relating to various views of an engine which are stored on the example.org web site (`http://example.org/webcgm/`). The CGM files contain various views of an engine assembly - the top view, the front view, the right view, the left view, and the isometric view. The CGM files are identified as follows:

Metafile 1: "engine_top.cgm"

Metafile 2: "engine_front.cgm"

Metafile 3:"engine_right.cgm"

Metafile 4:"engine_left.cgm"

Metafile 5: "engine_iso.cgm"

Each metafile contains one picture, with picture id identical to the metafile name (without the ".cgm"), and with several identifiable objects — the oil pump, the cylinder head, the fan, the radiator, or the distributor. Not all objects are shown in all views.

The objects contained in each metafile are as follows:

Metafile 1:

Oil pump: id='oil-pump-t' name='lube-system'

Cylinder head: id='cyl-hd-t' name='engine'

Fan: id='fan-t' name='cooling'

Radiator: id='rad-t' name='cooling'

Distributor: id='dist-t' name='ignition'

Metafile 2:

Oil pump: id='oil-pump-f' name='lube-system'

Cylinder head: id='cyl-hd-f' name='engine'

Fan: id='fan-f' name='cooling'

Radiator: id='rad-f' name='cooling'

Distributor: id='dist-f' name='ignition'

Metafile 3:

Oil pump: id='oil-pump-r' name='lube-system'

Cylinder head: id='cyl-hd-r' name='engine'

Fan: id='fan-r' name='cooling'

Radiator: id='rad-r' name='cooling'

Metafile 4:

Cylinder head: id='cyl-hd-l' name='engine'

Fan: id='fan-l' name='cooling'

Radiator: id='rad-l' name='cooling'

Distributor: id='dist-l' name='ignition'

Metafile 5:

Oil pump: id='oil-pump-i' name='lube'

Cylinder head: id='cyl-hd-i' name='engine'

Fan: id='fan-i' name='cooling'

Radiator: id='rad-i' name='cooling'

Distributor: id='dist-i' name='ignition'

3.1.3.2 Example 1

1st param: `http://example.org/webcgm/engine_top.cgm#pictseqno(1).id(cyl-hd-t,full+newHighlight)`
3rd param: `_blank`

When used as the value of the 'linkuri' APS Attribute (1st and 3rd parameters) in an object in a CGM file, this example retrieves engine_top.cgm CGM file from the example.org web site and displays the first (only) picture in a new window, highlighting the object with an id of "cyl-hd-t" in a full-picture view. The existing display window remains unchanged. The above expressions represent the preferred form (see 3.2.2.3) when the link is contained within CGM content. The following form is legal, but discouraged (may be removed from future version of WebCGM):

`http://example.org/webcgm/engine_top.cgm#pictseqno(1,_blank).id(cyl-hd-t,full+newHighlight)`

3.1.3.3 Example 2

`http://example.org/webcgm/engine_top.cgm#pictid(engine_top).id(oil-pump-t,full+newHighlight)`

When used as the IRI in an OBJECT element in HTML, this example displays the CGM inside a rectangle defined by the width and height parameters of the OBJECT tag, displaying the whole picture with the pump highlighted.

3.1.3.4 Example 3

IRI: `http://example.org/webcgm/engine_iso.cgm#id(dist-i,zoom+newHighlight)`
HTML 'target' attribute: `topframe`

When used as the value of the IRI to target an object in a CGM file from an HTML file, this IRI (plus HTML attribute) retrieves the engine_iso.cgm CGM file from the example.org web site and displays the picture in the metafile in the frame named "topframe", highlighting the object with an id of "dist-i". If present, the 'viewcontext' APS Attribute for the object "dist-i" is used to determine the rectangular portion of the picture to display in the frame, else the fallback computation of a target rectangle used.

3.1.3.5 Example 4

1st param: `http://example.org/engine_front.cgm`
3rd param: `topframe`

When used as the value of the 'linkuri' to target an object in a CGM file from another CGM file, this linkuri (1st and 3rd parameters) retrieves the engine_front.cgm CGM file from the example.org web site and displays the picture in the metafile in the frame named "topframe", with a full-picture view. (See example 1, about alternate but discouraged forms.)

3.1.3.6 Example 5

```
http://example.org/webcgm/engine_top.cgm#name(cooling)
```

This example retrieves the engine_top.cgm CGM file from the example.org web site and displays the picture in the metafile. The view zooms to the target rectangle, which contains all objects with 'name' APS Attribute with value "cooling", and each such object is highlighted.

3.1.3.7 Example 6

```
http://example.org/webcgm/engine_top.cgm#fan-t
```

This example retrieves the engine_top.cgm CGM file from the example.org web site and displays the first (only) picture in the metafile. The view is zoomed to the computed target rectangle of the object with id "fan-t", and it is highlighted.

3.1.3.8 Example 7

```
#id(oil-pump-t, addHighlight)
```

This example will leave unchanged the current zoom and pan factors in the currently displayed picture, and will highlight the `oil-pump-t` object. Any existing highlighting of other objects in the picture is preserved.

3.2 Application Structure and APS Attribute descriptions

3.2.1 Application Structures

WebCGM defines these Application Structure (APS) types: layer, grobject, para, subpara, and grnode. This document uses the term "object" to refer to an APS of type 'grobject', 'para', and 'subpara'.

Although the picture body of a WebCGM picture is not itself an APS, the content rules of the picture body (`picbody`) are defined by this piece of EBNF (see the fragment syntax for definition of notation):

```
picbody ::= layer+ |
            (grobject | para | grnode | gdata)*
```

I.e., a picture body contains either one or more layers, or else it contains a collection of eligible APSs ('grobject', 'para', 'grnode') and graphical data (`'gdata'`). The content rules of these APSs are defined in the following sections.

CGM:1999 requires that the Begin APS element of every Application Structure have a unique identifier parameter. The character repertoire of the APS id parameter in WebCGM content is identical to the repertoire defined for the `objid` fragment production in Section 3.1.1.3.

3.2.1.1 Grobject

Description. The Application Structure (APS) of type 'grobject' is used to group graphical primitives in a picture together and assign certain attributes to the group. The object is geometrically identified either by the set of primitives enclosed between the BEGIN APS BODY and END APS elements (if any), or by the spatial region associated with the 'region' APS Attribute (if present). APSs of type 'grobject' may contain any CGM graphical content allowed by this profile.

Definition: The **interactive region** of an object is the effective geometric region for the purposes of all interactive cursor and mouse operations, such as picking and mouseover. By default, the *drawn graphical primitives* of the object define the interactive region. For filled-area primitives this includes: the edge, if edge visibility is 'on'; the interior, if the interior style is other than 'empty' or 'hollow'; and, the boundary, for interior style 'hollow'. For all graphical primitive types, *drawn graphical primitives* exclude any that are fully transparent (so a fully transparent object is equivalent to an empty object, for purposes of interactive region definition). If the object contains a 'region' APS Attribute, then that region area is the interactive region.

Content Model. Except when occurring in Text Open State (see 3.2.1.7), the content of an APS of type 'grobject' is (see fragment syntax for definition of EBNF notation)::

- `grobject ::= (grobject | para | grnode | gdata)*`
- grobject APS may also contain optional APS attributes of type: region, viewcontext, linkuri, screentip, name, visibility, interactivity.

Viewer Behavior. The selection ("pick") of a 'grobject' APS, as well as other objects (APSs) that may be the target of a "pick" event, follow the rules of WebCGM DOM events. Those rules determine all aspects of the event processing, including the selection of the proper target object when there are multiple eligible candidates, and the selection of the proper handler to process the event. Viewers shall give visual feedback to the user that a successful pick has occurred, and an indication of the particular object (or region) which has been picked. The exact method of feedback is viewer dependent.

> **Example**. A common example in WebCGM usage scenarios is a simple 'grobject', that contains a linkuri APS Attribute. In this simple case, if an appropriate mouse event handler on the Metafile does not handle the event and prevent further processing, the WebCGM DOM event model says that the event will be "passed on for hyperlink processing." The event model dictates the following outcomes for the common cases:
>
> - if the mouse click is in the interactive region of only one APS, it is "picked", and its 'linkuri' is executed;
> - if the mouse click is in the interactive region of more than one APS, the topmost APS (latest occurring in the metafile) is the one that is "picked", and its linkuri is executed.
>
> If, on the other hand, the object (APS) that is "picked" by the event model rules does not contain a linkuri, then no hyperlink processing occurs and the event is passed along for further processing.

If an APS is the target of a link, either from within the picture or from content external to the picture, then the behavior of the viewer shall be as defined in the section "Object behaviors".

The CGM:1999 standard allows the definition of an APS to be continued in pieces which are disjoint in the file. If an APS occurs which has the same value of the 'id' parameter as an earlier APS occurrence, then that is construed as a continuation of the definition of that object. Since version 2.0, continued APS constructs are prohibited in WebCGM metafile instances.

3.2.1.2 Layer

Description. The 'layer' APS declares that the graphical content within this APS and any valid nested APS ('grobject', 'para', and 'grnode', but not 'layer') belongs to the layer identified by the contained 'layername' APS attribute.

Content Model. The content of an APS of type 'layer' is (see fragment syntax for definition of EBNF notation):

- `layer ::= (grobject | para | grnode | gdata)*`
- exactly one APS attribute of type: layername
- layer APS may also contain optional APS attributes of type: layerdesc, visibility, interactivity.

Viewer Behavior. Viewers shall provide functionality to inform users of the presence of layers, their names and descriptions. Viewers shall provide functionality to selectively turn on and off the visibility of layers. Viewers may, but are not required to, provide additional functionality for the view manipulation and browsing of layers.

3.2.1.3 Para

Description. The application structure (APS) of type 'para' may be used to identify text ("paragraphs"). 'Para' together with 'content' can potentially enable applications to build text search functionality, especially in cases where the underlying graphical data does not comprise graphical text in a searchable form (e.g., the text has been rasterized, polygonized, or visually-single strings are fragmented into multiple smaller text elements). Except when occurring in Text Open State, 'para' APSs may contain any CGM graphical content allowed by this profile.

Content Model. Except when occurring in Text Open State (see 3.2.1.7), the content of an APS of type 'para' is (see fragment syntax for definition of EBNF notation):

- `para ::= (subpara | gdata)*`
- para APS may also contain optional APS attributes of type: region, viewcontext, linkuri, screentip, name, content, visibility, interactivity.

Viewer Behavior. With respect to object selection and link navigation, the viewer behavior of 'para' is identical to that of 'grobject' (3.2.1.1).

This version of WebCGM does not standardize text search functionality, but provides facilities with which applications can build such functionality. It is anticipated that a future version will define standard search functionality.

> **Example.** A search match priority that applications might use (originally recommended in WebCGM 1.0) is: 'para' with matching 'content' (1st priority match); 'para' without 'content' but with recognizable single-element RESTRICTED TEXT match (2nd priority match); or, single-element RESTRICTED TEXT match, outside of any 'para' (3rd priority match).

3.2.1.4 Subpara

Description. The application structure (APS) of type 'subpara', may be used to identify smaller fragments of text within APS of type 'para'. This enables, for example, the identification of the larger text block (the "paragraph") for searching purposes, and the tagging of smaller fragments as hotspots. Except when occurring in Text Open State (see 3.2.1.7), 'subpara' APSs may contain any CGM graphical content allowed by this profile. 'Subpara' APS may not contain any nested APS.The APS attribute content rules of sub-para matches those of 'para'.

Content Model. Except when occurring in Text Open State (see 3.2.1.7), the content of an APS of type 'subpara' is (see fragment syntax for definition of EBNF notation):

- `subpara ::= (gdata)*`
- subpara APS may also contain optional APS attributes of type: region, viewcontext, linkuri, screentip, name, content, visibility, interactivity.

Viewer Behavior. See 3.2.1.3, 'para'.

3.2.1.5 Grnode

Description. The application structure (APS) of type 'grnode' is meant to group basic graphical primitives only. For this reason, 'grnode' must contain the APS 'id' parameter required by CGM:1999 for all APS, but a 'grnode' cannot contain any APS attribute elements. The content of a 'grnode' is, however, not limited to graphical primitives. The allowed APS content of a 'grnode' is the same as for the 'grobject' Application Structure.

Even if 'visibility' and 'interactivity' are not allowed on the APS, the 'grnode' supports inheritance (i.e., it is possible to make a 'grnode' visible or non-visible by inserting it within an object which support the 'visibility' attribute).

Content Model. The permissible content of an APS of type 'grnode' is:

- `grnode ::= (grobject | para | grnode | gdata)*`

Viewer Behavior. Unlike other application structures, 'grnode' is not interactive; i.e., it does not receive mouse events. If a mouse event is triggered on the geometry of a 'grnode', an ancestor node of type 'grobject' may respond to the event. Therefore, the content of a 'grnode' could effectively appear to be interactive, for example, if the 'grnode' were a direct child of a 'grobject'. See the Event interface for more information regarding mouse events. APS of type 'grnode' also do not support most DOM functionality such as Style Properties, object-extent inquiry, etc. See the appropriate sections of the DOM chapter for details.

3.2.1.6 About general metadata elements

Description. This version and level of WebCGM do not allow additional APS elements to occur, other than 'grobject', 'layer', 'para', 'subpara', and 'grnode'. Private metadata may be associated with WebCGM objects by keeping the metadata outside of the CGM, and associating it with objects within the CGM. A means for binding external private metadata to WebCGM instances is defined in the section *XML Companion File* (XCF).

3.2.1.7 APS in Text Open State

The rules of CGM:1999 (corrected) allow an APS to occur in Text Open State, i.e., after a 'notfinal' RESTRICTED TEXT (RT) element but before the terminating 'final' APPEND TEXT (AT) element. This allows, for example, a substring comprising one of the interior APPEND TEXT elements in a RT-AT-AT-...-AT sequence to be contained within an APS. Thus, for example, the APPEND TEXT element (substring) could be the target of a link, or could be the source of a link (a "hotspot").

Such an APS is referred to as a *substring APS*.

WebCGM 2.1 restricts the APS types of such substring APS to: 'grobject', 'para', and 'subpara'. For APS of these types that are *not* substring APS, the content rules are as quoted in sections 3.2.1.1, 3.2.1.3, and 3.2.1.4 respectively. If an APS of one of these types is a substring APS, then its content is restricted as follows by the rules of CGM:1999 (corrected): only the APPEND TEXT element and those text attributes normally allowable in Text Open State (under CGM:1999 rules).

The allowable APS Attributes for 'grobject', 'para', and 'subpara' are as designated in 3.2.1.1, 3.2.1.3, and 3.2.1.4 respectively, regardless of whether or not they are substring APS.

The WebCGM DOM (Chapter 5) and WebCGM XCF (Chapter 4) allow the application of geometric transforms to APS of all types except 'grnode'. Therefore, a geometric transform could potentially be targetted at an APS ('grobject', 'para', or 'subpara') that contains a substring APS. In this case, the layout and rendering rules of CGM:1999 for Restricted Text and Append Text (see ISO/IEC 8632:1999, section 6.7.3) shall have precedence. Any attempt to set a geometric transform on a substring APS shall have no effect.

3.2.2 Application Structure Attributes

3.2.2.1 Region

Initial value: none
Applies to: 'grobject', 'para', 'subpara'
Inherited: no

Description. The 'region' APS Attribute provides an optional spatial region, associated with a graphical object, which assumes precedence for mouse event operations directed at the object, such as picking, mouseover, etc. This APS Attribute, if present, defines the *interactive region* of an object; else, the interactive region is defined by the geometry of the object.

Simple regions of type rectangle, ellipse, polygon, and polybezier can be defined. Complex regions which comprise a collection of simple regions can be built, allowing definition of disjoint subregions, regions with holes, etc. Their semantics (subregions, and interior/exterior definition) are identical to those of the CGM element CLOSED FIGURE. At most one 'region' attribute may be present within a single APS.

Parameters. The data record is an SDR of one or more member pairs (i.e., 2*m members, m>=1). Each

member-pair defines a simple region: the first member is of data type Index, whose valid values are:

1. rectangle
2. ellipse
3. polygon
4. continuous polybezier

The second member is type VDC and contains:

- for rectangle: 4 VDC defining two corner points;
- for ellipse: 6 VDC defining respectively the center, and two CDP endpoints;
- for polygon: 2n VDC defining polygon vertex points
- for polybezier: 2*(3n+1) VDC values, representing 3n+1 points, defining n contiguous cubic bezier segments;

For polygon and polybezier regions, closure is implicit (if the last given point does not match the first, then the viewer closes the region with a straight line segment from the last to the first).

In the case that there are multiple simple regions, m>1, then the individual simple regions each correspond to a REGION in the sense of the CGM CLOSED FIGURE element.

Viewer Behavior. See 3.2.1.1.

3.2.2.2 Viewcontext

Initial value: none
Applies to: 'grobject', 'para', 'subpara'
Inherited: no

Description. The 'viewcontext' APS Attribute provides a specification to viewers of the initial view of an object, when the viewer has been directed to navigate to the graphical object which contains this attribute. A 'viewcontext' APS Attribute may be contained within an otherwise empty APS, in which case the APS provides only a viewport specification. At most one 'viewcontext' attribute may be present within a single APS.

Parameters. The data record is an SDR of 1 member of type VDC defining two corner points of a rectangle.

Viewer Behavior. See 3.2.1.1.

3.2.2.3 Linkuri

Initial value: none
Applies to: 'grobject', 'para', 'subpara'

Inherited: no

Description. The 'linkuri' APS Attribute defines a IRI, to be associated with the object containing this attribute. When the object is selected by a graphical pick operation, and the WebCGM event model determines that hyperlink processing shall handle the event, then the viewer shall take necessary action to navigate the link. Multiple 'linkuri' attributes may be contained within a single APS. If the object contains more than one 'linkuri' attribute, the user shall be given a choice of which IRI to navigate.

Parameters. The data record is an SDR of one member, containing three strings (type SF, String Fixed). The first string is the link destination, a IRI, the second string (possibly null) is a Link Title parameter, and the third string (possibly null) is the Behavior parameter. Note that a null string is a zero-length string, and is not the same as an omitted parameter. The parameter must not be omitted.

The destination of a link is specified by a Internationalized Resource Identifier, or IRI. See section 3.1.1.4 for further discussion of this parameter. This specification does not constrain the syntax or semantics of a IRI in a 'linkuri' that identifies a resource that is not a CGM file (for example, an HTML or XML document).

The Behavior string defines picture behavior associated with the link. The values and meanings are as defined in 3.1.2.2. In cases that the destination is not CGM media type, the 3rd parameter, Behavior, shall be used if picture behavior is to be specified for the link (there is no other option). The Behavior string may also be used for links to CGM media types, and is the preferred method.

In the case that the IRI points to CGM media type, the picture behavior may be encoded within the optional fragment identifier in conjunction with the IRI structure, per section 3.1.1, " IRI fragment specification". This form is discouraged, and may be removed in a future edition of this profile. For specifying picture behavior, particular WebCGM 'linkuri' instances shall use either the Behavior string (preferred), or the picture behavior specification embedded in the fragment (discouraged), but not both.

EXAMPLES. The following example illustrates the only allowable form of a CGM-to-HTML link that would open an HTML document in a new, blank window and navigate to an anchor, `myAnchor`, in the document:

1. `myHTMLfile.html#myAnchor` as 'linkrui' 1st parameter, plus 'linkuri' 3rd parameter value of `_blank` .

The preferred form of an analogous CGM-to-CGM link — opening a CGM in a new blank window, and navigating to a particular object — is shown in the following two examples:

2. `myCGMfile.cgm#picseqno(1).objid(someId,zoom)` , plus 'linkuri' 3rd parameter value of `_blank,`
3. or simply: `myCGMfile.cgm#objid(someId,zoom)` , plus 'linkuri' 3rd parameter value of `_blank` .

The following example illustrates an allowed, but discouraged, variant of the forms #2 and #3:

4. `myCGMfile.cgm#pictseqno(1,_blank).objid(someId,zoom)`

3.2.2.4 Layername

Initial value: none
Applies to: 'layer'
Inherited: no

Description. The 'layername' APS Attribute declares that the graphics associated with the 'layer' APS containing this attribute belong to the identified layer. The 'layername' is not required to be unique. If more than one 'layer' APS contains the same 'layername', then the occurrences following the first occurrence shall be construed as continuing the definition of the named layer. Exactly one 'layername' attribute must be present within each 'layer' APS.

Parameters. The data record is an SDR of one member, containing one string (type SF, String Fixed) - the Layer Name (identifier). The string can be null (zero-length). If the Layer Name is null, then the graphics of this object belong to the null layer.

Viewer Behavior. See 3.2.1.2.

3.2.2.5 Layerdesc

Initial value: none
Applies to: 'layer'
Inherited: no

Description. The 'layerdesc' APS Attribute provides optional descriptive text which is associated with the 'layer' APS in which it occurs. This may be used by viewers to facilitate required and optional layer manipulation functions, as described in 3.2.1.2. At most one 'layerdesc' attribute may be contained within a single 'layer' APS.

Parameters. The data record is an SDR of one member, containing one string (type SF, String Fixed).

Viewer Behavior. See 3.2.1.2.

3.2.2.6 Screentip

Initial value: none
Applies to: 'grobject', 'para', 'subpara'
Inherited: no

Description. The 'screentip' APS Attribute provides an optional string, to be associated with a graphical object, which viewers can display when the graphical cursor passes over the *interactive region* of the graphical object. Whether or not the screentip display actually occurs depends on how the WebCGM event model determines that mouse-over events should be handled. This APS Attribute may occur within any graphical object of WebCGM, specifically, within any APS of type 'grobject', 'para', and 'subpara', and there shall be at most one occurrence within any particular APS.

Parameters. The data record is an SDR of one member, containing one string (type SF, String Fixed).

Viewer Behavior. Viewers shall be capable of displaying the screen tip, if one is defined for a graphical object, visible to the user when the cursor passes over the graphical object, in the common style of Web browsers.

3.2.2.7 Name

Initial value: none
Applies to: 'grobject', 'para', 'subpara'
Inherited: no

Description. The 'name' APS Attribute provides an optional string, that defines a "common name" associated with an object. Unlike the APS 'id' parameter, the 'name' APS attribute need not be unique within a metafile. Multiple 'name' attributes may be contained within a single APS.

Parameters. The data record is an SDR of one member, containing one string (type SF, String Fixed). The character repertoire of the name APS attribute in WebCGM content is identical to the repertoire defined for the `objname` fragment production in Section 3.1.1.3.

Viewer Behavior. The 'name' gives applications a way to associate common names with objects. The object can optionally be addressed by the value of the 'name' attribute. See 3.1.1.

3.2.2.8 Content

Initial value: none
Applies to: 'para', 'subpara'
Inherited: no

Description. The 'content' APS Attribute provides a means to declare what is text content in a 'para' APS and in a 'subpara' APS. It is provided as a basis on which applications can build text search (which is not further standardized in this version of WebCGM). Text that is apparent in a graphical display may not correspond to recognizable text strings in the metafile content itself. For example, the metafile content may draw the text with raster elements, or with filled strokes, or one character at a time in random order, etc.

A 'content' attribute on a 'para' APS should contain all of the rendered text of the 'para' APS, and a 'content' attribute on a 'subpara' APS should contain all of the rendered text of the 'subpara' APS. The 'content' APS Attribute may occur only within APS of type 'para' and 'subpara', and there shall be at most one occurrence within any such APS.

Parameters. The data record is an SDR of one member, containing one string (type SF, String Fixed).

Viewer Behavior. See the description under the 'para' APS, 3.2.1.3.

3.2.2.9 Visibility

Initial value: on
Applies to: 'layer', 'grobject', 'para', 'subpara'
Inherited: yes

Description. The 'visibility' attribute indicates if an object is visible or not. 'Visibility' applies to Application Structures (APS) of type 'layer', 'grobject', 'para' and 'subpara'. The value of 'visibility' is inherited by any descendant objects of type 'layer', 'grobject', 'para', and 'subpara'. Although it can't be set on 'grnode' and doesn't apply to 'grnode', 'visibility' also inherits to objects of type 'grnode', and inherits from objects of type 'grnode' to any descendents in the WebCGM structure.

Parameters. The data record is an SDR of one member, containing one string (type SF, String Fixed). The valid values are 'off', 'on', 'inherit'.

Viewer behavior. A non-visible object is not displayed. A non-visible object behaves like a non-interactive object (i.e., it cannot be clicked or highlighted). This does not imply that the 'interactivity' attribute is changed to off, but simply that the user agent must not respond to mouse events.

3.2.2.10 Interactivity

Initial value: on
Applies to: 'layer', 'grobject', 'para', 'subpara'
Inherited: yes

Description. The 'interactivity' attribute indicates if an object may receive mouse events. 'Interactivity' applies to Application Structures (APS) of type 'layer', 'grobject', 'para' and 'subpara'. The value of 'interactivity' is inherited by any descendant objects of type 'layer', 'grobject', 'para', and 'subpara'. Although it can't be set on 'grnode' and doesn't apply to 'grnode', 'interactivity' also inherits to objects of type 'grnode', and inherits from objects of type 'grnode' to any descendents in the WebCGM structure.

Parameters. The data record is an SDR of one member, containing one string (type SF, String Fixed). The valid values are 'off', 'on', 'inherit'.

Viewer behavior. When the 'interactivity' of an object is set to off, events for this object are disabled. This

has the effect of disabling event handlers, cursor changes, highlighting, screentip and hyperlinking for the given node and its descendant. An object that is the target of a link always responds to highlighting, regardless of its 'interactivity' attribute value.

3.2.2.11 About general metadata elements

Description. This version and level of WebCGM do not allow additional APS Attribute elements to occur, other than as enumerated above. Private metadata may be associated with WebCGM objects by keeping the metadata outside of the CGM, and associating it with objects within the CGM. A means for binding external private metadata to WebCGM instances is defined in the section *XML Companion File* (XCF).

3.3 Content model

This subsection is informative (non-normative).

This is an informative, at-a-glance summary of the whole content model of the CGM Version 4 functionality of WebCGM — the "Intelligence" content — using the formal specification technique of the XML DTD. It has been suggested that validating XML parsers could be adapted to perform content validation of WebCGM instances (either via modification of the readers, or via transformation of the intelligent content of the WebCGM instance).

```
<!-- To document the structure of the CGM Version 4      -->
<!-- content of WebCGM the following DTD fragment        -->
<!-- has been developed.                                 -->
<!--                                                     -->
<!-- PICBODY is included in this DTD fragment for        -->
<!-- purposes of demonstrating that the layer, grobject, -->
<!-- and para structures can exist within the picture    -->
<!-- body level in a CGM instance. The gdata element     -->
<!-- with its associated cgmprim entity attribute is     -->
<!-- intended to represent the model for CGM data stored -->
<!-- as an external entity.                              -->
<!--                                                     -->
<!-- Note: of the attributes listed below, all           -->

<!-- correspond to APS Attribute elements of CGM, except -->
<!-- the 'ID', which corresponds to the 'id' parameter   -->
<!-- of the Begin Application Structure CGM element.     -->


<!ELEMENT picbody (layer+ | (grobject | para | grnode
                             | gdata)*)                    >

<!ELEMENT layer (grobject | para | grnode | gdata)*        >
<!ATTLIST layer
```

```
  id            ID         #REQUIRED
  layername     CDATA      #REQUIRED
  layerdesc     CDATA      #IMPLIED
  visibility    (on | off | inherit) #IMPLIED
  interactivity (on | off | inherit) #IMPLIED
                                                    >
<!ELEMENT grobject (grobject | para | grnode | gdata)*     >

<!ATTLIST grobject
  id            ID         #REQUIRED
  region        CDATA      #IMPLIED
  viewcontext   CDATA      #IMPLIED
  linkuri       CDATA      #IMPLIED
  screentip     CDATA      #IMPLIED
  name          CDATA      #IMPLIED
  visibility    (on | off | inherit) #IMPLIED
  interactivity (on | off | inherit) #IMPLIED
                                                    >
<!ELEMENT para (subpara | gdata)*                     >
<!ATTLIST para
  id            ID         #REQUIRED
  region        CDATA      #IMPLIED
  viewcontext   CDATA      #IMPLIED
  linkuri       CDATA      #IMPLIED
  screentip     CDATA      #IMPLIED
  name          CDATA      #IMPLIED
  content       CDATA      #IMPLIED
  visibility    (on | off | inherit) #IMPLIED
  interactivity (on | off | inherit) #IMPLIED
                                                    >
<!ELEMENT subpara (gdata)*                            >
<!ATTLIST subpara
  id            ID         #REQUIRED
  region        CDATA      #IMPLIED
  viewcontext   CDATA      #IMPLIED
  linkuri       CDATA      #IMPLIED
  screentip     CDATA      #IMPLIED
  name          CDATA      #IMPLIED
  content       CDATA      #IMPLIED
  visibility    (on | off | inherit) #IMPLIED
  interactivity (on | off | inherit) #IMPLIED
                                                    >
<!ELEMENT grnode (grobject | para | grnode | gdata)*      >
<!ATTLIST grnode
  id            ID         #REQUIRED
                                                    >
<!ELEMENT gdata EMPTY                                 >
<!ATTLIST gdata
```

```
cgmprim       ENTITY     #REQUIRED                          >
```

Note: the use of XML to express the content model of WebCGM implies that a particular attribute can have at most one instance within a particular APS instance. This is not the case, and the normative rules are as specified in 3.2.2.1 through 3.2.2.10.

3.4 WebCGM and the `object` element

The only standard way to reference inline CGMs from HTML documents is through the `object` element, using the `data` attribute for the CGM file and the `type` attribute to specify the full Mime Type. The minimal element for adding CGM into a document would be:

```
<object data="xxx.cgm" type="image/cgm;Version=4;ProfileId=WebCGM"
width="200" height="100" />
```

Other HTML 4.01 attributes which may be used on the `object` tag include `align`, `border`, `hspace`, `id`, `name` and `vspace`. Use of `align`, `border`, `hspace`, and `vspace` is only permitted in Transitional HTML 4.01, not Strict HTML 4.01.

The attributes `classid`, `codebase`, `declare`, `shapes`, `usemap`, `codetype`, and `archive` are prohibited. Content which uses WebCGM shall not make direct reference to the code which may be used to display it.

The event-related attributes, ONCLICK,...,ONMOUSEOVER, are permitted but their effect is undefined in this version of WebCGM. Instead the WebCGM DOM offers an addEventListener method for adding event listeners on WebCGM documents

The attributes `accesskey`, `alt`, `class`, `dir`, `lang`, `longdesc`, `standby`, `style`, `tabindex` and `title` are permitted, but have no defined effect on CGM viewers and display of CGM pictures. They are used to improve accessibility, and may also affect the presentation of any alternative text content of the `object` element.

The `object` element can contain optional `param` elements, which allow the HTML to pass additional data to the target object. The following `param` elements are defined and permitted for WebCGM. Each `param` is presented as a name followed by permissible values (after the ":"), and description. The names and the permissible values are case-insensitive, with the single exception of the value of the `'onload' param` (which identifies the event handler script function that is to be invoked upon the `onload` event, represented below as"`<eventHandlerName(evt)>`".

onload: <eventHandlerName(evt)>

The user supplies the name of an event handler that the viewer shall invoke upon the 'onload' event. This `param` element and handler allows for script writers to manipulate the WebCGM DOM

at the point at which the user agent has fully parsed the `object` tag and its descendants and is ready to render the object to the screen.

EXAMPLE:

```
<script type="application/ecmascript"> <![CDATA[
function myHandler(evt)
{
// performs DOM manipulation calls...
}
]]>
</script>
[...]

<object data="xxx.cgm" type="image/cgm;Version=4;
ProfileId=WebCGM"
                        width="200" height="100">
  <param name="onload" value="myHandler(evt);"/>
</object>
```

fixed: No | Yes

Disables ("Yes") or enables ("No") the user zoom and pan controls of the viewer. It does not affect picture navigation by other means, such as object behaviors in IRI fragments. Default is No. For the value Yes, viewers shall not put up scroll bars and zoom options, which would normally be offered.

background: Enable | Disable

States whether the CGM picture shall be drawn with its normal background color, as given in the CGM picture (Enable), or whether the background color of the picture shall be suppressed (Disable), thus allowing the background color or background image of the HTML page to show through. Default is Enable. Note: the background color of the picture is either default, or is explicitly defined by the BACKGROUND COLOUR element in the CGM picture. (See section 2.2.3.)

The `background` param element is obsolete as of version 2.1 of this profile.

viewport: topx topy botx boty

Gives a viewport of the CGM (a part of the picture) to display. The values are the top-left and bottom-right corners of the sub-picture. The units are either in the Virtual Device Coordinates (VDC) of the CGM; or, as a percentage of the picture's VDC EXTENT element (whether explicit or default), if the value is followed by a percent sign (%). This facility allows for different parts of a CGM picture to be displayed at different scales in different places in the document. Default is full VDC EXTENT. Note: the use of `viewport` can conflict with some options (e.g., those object behaviors which effectively select a sub-picture via a 'viewcontext' APS attribute within the CGM picture) in a IRI fragment on the `data` attribute of the `object` element. In case of conflict, the `viewport` specification shall have precedence.

The `viewport` param element is deprecated and may be removed in a future version of this profile.

mapping: fit | fill
halign: top | middle | bottom
valign: left | middle | right

Each CGM picture has a fixed aspect ratio, determined by the VDC EXTENT element, which may not agree with the aspect ratio defined by `height` and `width` specified on the `object` tag. These three parameters can be used to specify where and how to place the CGM within the window specified on the `object` tag. 'Fit' specifies to isotropically scale the picture (or sub-picture) so that one dimension fits exactly in the window — there will be undrawn space left in the window in one dimension if the aspect ratios don't match. In this case, the handling of the extra space is defined by the values of 'halign' (if the extra space is in the horizontal dimension) and 'valign' (if the extra space is in the vertical dimension). 'Fill' means to isotropically scale so that the window is filled in both directions — if the aspect ratios don't match, a part of the picture will be clipped away at the window boundary. In this case, 'halign' and 'valign' define what part of the picture is shown (and what part is clipped away) in the dimension that needs to be clipped.

The default values are 'fit', 'middle', 'middle'.

These `param` elements differ from the `align` attribute of `object`, which is used to specify where the `object` is placed in the document. This could be expressed using the `param` element as:

```
<param name="halign" value="middle" />
<param name="valign" value="middle" />
```

Figures 5 and 6 show the different effects achieved by 'Fit' and 'Fill' with the different alignments.

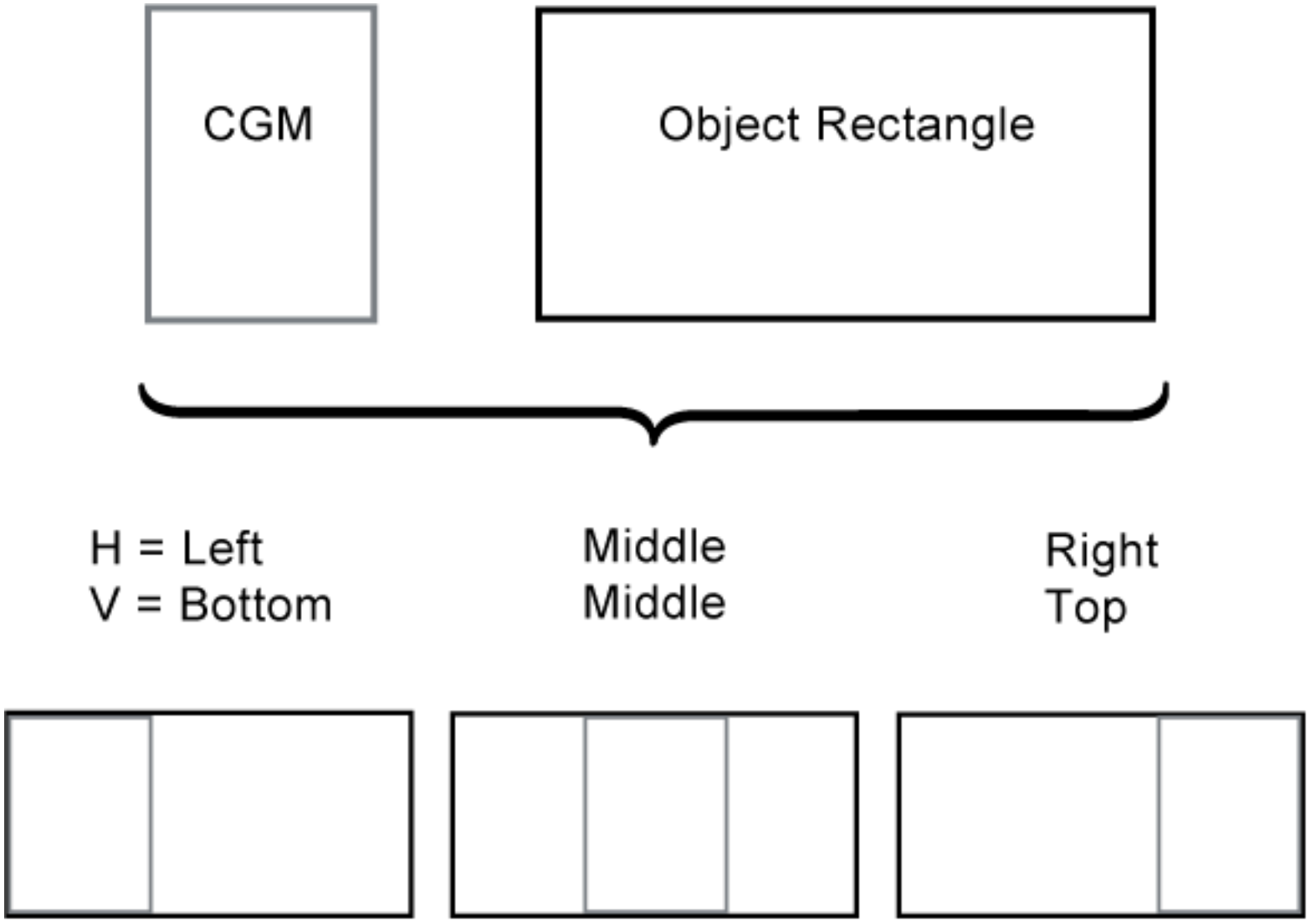

Figure 5. Effects of fit

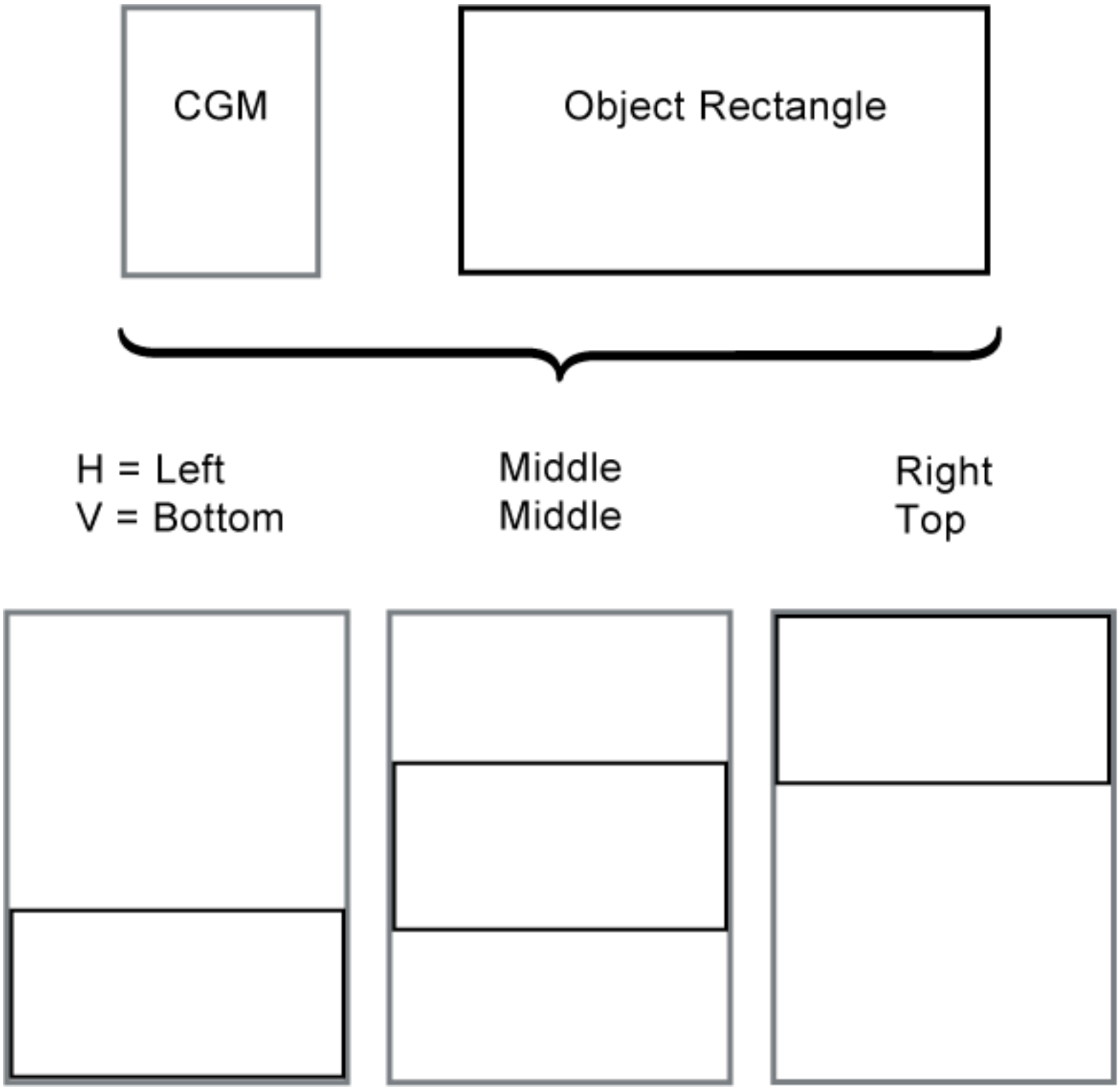

Figure 6. Effects of fill

Back to top of chapter

---

4. WebCGM XML Companion File

This chapter and its sections are normative, unless otherwise indicated.

Contents

4.1 Introduction & examples

This section and its subsections are informative (non-normative).

The WebCGM XML Companion File (XCF) was added to WebCGM in the 2.0 version, and further enhanced in this 2.1 upgrade. The element and attribute definitions found in this section represent the WebCGM XCF DTD. This DTD may be extended by profiles deriving from WebCGM. The WebCGM XML companion files may be used for several purposes. There are many conceivable usage scenarios, but for the scope of WebCGM, the following four were identified as most important.

Scenario 1: A companion file can be used to bind application specific details (such as a part number) to a particular Application Structure. It is up to the application to control how the application-specific information is used.

Example 4.1: *A companion file used to relate some application-specific data to graphical objects*.

```
<!ENTITY % grobjectAttEXT "model:partNum CDATA #IMPLIED" >
<webcgm version="2.1" id="root-cgm" filename="sample_1.cgm"
                      xmlns:model="http://example.org"
                      xmlns="http://www.cgmopen.org/schema/webcgm/">
<grobject apsid="id_1" model:partNum="bolt-100A"/>
<grobject apsid="id_2" model:partNum="wingnut-T9"/>
...
<grobject apsid="id_49" model:partNum="drill-bit-D01"/>
<grobject apsid="id_50" model:partNum="router-bit-B389"/>
</webcgm>
```

Scenario 2: A companion file could also be used to update a CGM illustration via the WebCGM DOM (see section *Relationship with XML companion file* for more information):

Example 4.2: *A companion file used to update sample_2.cgm before being displayed by a user agent. Calls to some WebCGM DOM methods need to take place to perform this task*.

```
<webcgm version="2.1" id="root-cgm" filename="sample_2.cgm"
                      xmlns="http://www.cgmopen.org/schema/webcgm/">
<bindByName apstargetname="bolt_100" screentip="Replacement part:bolt-100B"/>
<bindByName apstargetname="wingnut_9" screentip="Replacement part:wingnut-T9A"/>
</webcgm>
```

Scenario 3: Although it is out-of-scope of this version of the WebCGM XCF to fully mirror the hierarchical structure of a CGM graphic (see "Structure overview"), an XML Companion File could be used as a partial, scaled down XML inventory of a CGM illustration by enumerating the Application Structures IDs, types and (most) attributes.

Example 4.3: *A companion file used as a partial inventory of sample_3.cgm. In this case, the description puts an emphasis on the hotspot region of each 'grobject' element*.

```
<webcgm version="2.1" id="root-cgm" filename="sample_3.cgm"
                      xmlns="http://www.cgmopen.org/schema/webcgm/">
<grobject apsid="id_1" region="1 0 0 100 100"/>
<grobject apsid="id_2" region="1 200 0 300 100"/>
...
<grobject apsid="id_49" region="1 1600 600 1700 700"/>
<grobject apsid="id_50" region="1 1800 600 1900 700"/>
</webcgm>
```

4.2 Content and structure of the XCF

This section and its subsections are normative, unless otherwise indicated.

4.2.1 Structure overview

It is not the intent of the WebCGM XML Companion File (XCF) to be a faithful XML representation of the object tree in a hierarchical WebCGM. Rather, XCF provides a mechanism to externalize both standardized and application (private) metadata from a structured WebCGM instance, and to bind it to the proper objects in the object tree of the WebCGM instance.

Accordingly, the structure of the XML Companion File is mostly flat. After the root element, `webcgm`, the various standard XCF elements occur as siblings in the companion file (with the single exception of `linkuri`). So, for example, `grobjects` that have a parent-child relationship in the CGM are siblings in the XCF. The normative WebCGM DTD for XCF expresses and enforces this flat structure, independent of whatever hierarchy may exist in the corresponding WebCGM instance.

Example 4.4: *A mostly-flat companion file binds standardized metadata to a hierarchical object tree in a WebCGM instance*.

```
<webcgm version="2.1" filename="sample_4.cgm"
                      xmlns="http://www.cgmopen.org/schema/webcgm/">
   <grobject apsid="level-4-obj" screentip="wingnut-400A"/>
   <grobject apsid="level-3-obj" screentip="bolt-assembly-100A"/>
</webcgm>
```

```
BegPic
  ..(graphics etc)..
  BegAps type="grobject" id="level-1-obj"
    BegAps type="grobject" id="level-2-obj"
      ..(graphics etc)..
      BegAps type="grobject" id="level-3-obj"
        BegAps type="grobject" id="level-4-obj"
          ..(graphics etc)..
        EndAps
      EndAps
    EndAps
  EndAps
EndPic
```

Note: Example 4.4 uses a condensed schematic representation of the CGM code, eliminating numerous details in order to illustrate the point.

Example 4.4 illustrates another point about the relationship of the XCF contents to the corresponding WebCGM instance — the order of the XCF is not required to follow the order of the CGM contents.

As suggested by Example 4.4, the root element of a conforming XCF instance must be the `webcgm` element. The `webcgm` element corresponds to the Picture object in the CGM. See *Relationship with XML Companion File* for more discussion.

The next section deals in detail with application-specific metadata attributes and elements in an XCF. In order to unambiguously establish where in the WebCGM object tree those metadata are to be inserted, the application-specific attributes and elements (defined in a separate namespace) are placed in the XCF as attributes and child elements on standardized XCF elements that correspond (bind) to the appropriate object in the WebCGM. This is seen in example 4.1, for application-specific attributes. See also section *Relationship with XML Companion File* for more discussion.

4.2.2 Extending the XML Companion File

The WebCGM DTD is extensible so that application-specific or industry-specific metadata may be added to the WebCGM object model (as shown in example 4.1). The extension definitions are implemented using namespaces. The DTD defines an extension entity for the content and attributes of most elements. As an example, a part manufacturer may want to associate parts information to graphical objects. This might be implemented with an extension that looks like:

```
<!ENTITY   % grobjectAttEXT   "model:partNum CDATA #IMPLIED" >
```

A host application could query the WebCGM DOM and retrieve the associated part information.

A set of rules must be followed when extending the WebCGM DTD:

1. The root element MUST be 'webcgm'.
2. The extended elements and/or attributes MUST be in another namespace (See Namespaces in XML).
3. Extending the list of children of an element MUST use the *elementName*EXT entity.

4. Extending the attribute list of an element MUST use the *elementName*AttEXT entity.
5. Elements from the WebCGM namespace MUST NOT be descendants of other namespaced elements.

The rules found above allow WebCGM user agents to process extended companion files in an interoperable manner.

4.2.3 The WebCGM XCF namespace

The WebCGM namespace:

```
http://www.cgmopen.org/schema/webcgm/
```

Namespace example:

```
<webcgm version="2.1" filename="sample.cgm" xmlns="http://www.cgmopen.org/schema/webcgm/">
```

Public Identifier for WebCGM 2.1:

PUBLIC "-//OASIS//DTD WebCGM 2.1//EN"

System Identifier for the WebCGM 2.1:

```
http://docs.oasis-open.org/webcgm/v2.1/webcgm21.dtd
```

DOCTYPE example. The following is an example document type declaration for a WebCGM XCF document:

```
<!DOCTYPE webcgm PUBLIC "-//OASIS//DTD WebCGM 2.1//EN" "http://docs.oasis-open.org/webcgm/v2.1/
webcgm21.dtd">
```

4.2.4 WebCGM XML Companion File conformance

A file is a conforming WebCGM 2.1 XCF document if it adheres to the specifications described in this (WebCGM 2.1) document, including those in the WebCGM 2.1 XCF DTD, and in addition all of the following conditions are met:

- it is a well-formed XML document (according to *XML1.0 3rd Edition*);
- its root element is `webcgm`
- if all non-WebCGM namespace elements and attributes and all **xmlns** attributes which refer to non-WebCGM namespaces are removed from the given document, and if an appropriate document type declaration (i.e., <!DOCTYPE webcgm ... >) which points to the WebCGM DTD is included immediately after the XML declaration (i.e., <?xml...?>), the result is a valid XML document.
- it conforms to Namespaces in XML, if any namespaces other than WebCGM are used in the document.

4.3 XCF elements

This section and its subsections are normative, unless otherwise indicated.

The standard XCF elements include:

- the standard root element, `webcgm`
- elements whose names match corresponding WebCGM object (APS) types — `layer`, `grobject`, `para`, `subpara`
- and a pair of general purpose metadata binding elements — `bindById` and `bindByName`.

The standard XCF elements also include:

- `linkuri`. Linkuri is an APS Attribute, but is encoded as an XCF element, rather than attribute, to avoid what would otherwise be an overly complex encoding of the string that would comprise its value as an XML attribute.

4.3.1 Data types and encoding

For the standardized XCF content, most of the items expressed on XCF elements as XML attributes have a straightforward correlation to a standardized WebCGM attribute or property that may be set or inquired with a WebCGM DOMcall.

In general, the encoding of XML attributes on XCF elements is identical to the encoding of the corresponding parameters in DOM calls. For example, the WebCGMAppStructure interface (section 5.7.6) defines 'viewcontext' as a simple string of four numbers, whitespace separated (see Wsp definition, section 5.5). The encoding of 'viewcontext' as an XML attribute on any allowed XCF element is the same as its encoding as a DOM method parameter.

Similarly, the Style Properties (settable on the WebCGMPicture and WebCGMAppStructure interfaces), as XML attributes on the XCF elements have the same valid values and are encoded identically as in the corresponding DOM calls.

Geometric transform, which is DOM-settable (see WebCGMAppStructure) for APS of all types except 'grnode', is realized in XCF as XML attributes on the eligible XCF elements. They have the same valid values as the corresponding DOM attributes and parameters. However, the Float and WebCGMMatrix types of the DOM are encoded into a WebCGMString XCF attribute value as List-of-number subtype.

EXAMPLE. The parameters of DOM setTransform() are a matrix and a 'replace' flag. The matrix is represented by six numbers (a, b, c, d, e, f). In XCF, the value of the setTransform attribute would be a WebCGMString: "a b c d e f combine". For example, this would scale an object about the origin by 0.5 and translate it right by 0.1:

```
<grobject apsid=someAPS" setTransform="0.5 0 0 0.5 0.1 0 combine" />
```

CAVEAT! In XML, one cannot assume the order of attributes on an element. So for example, one cannot assume that two consecutive geometric transforms on a <grobject> element will be applied in the order written.

EXAMPLE: one wishes to apply a translate followed by a rotate-about-origin to some APS.

Flawed, order of application is not guaranteed:

```
<grobject apsid=someAPS" translate="0.5 0.1 replace" rotate="45 0 0 combine" />
```

Correct:

```
<grobject apsid=someAPS" translate="0.5 0.1 replace" />
<grobject apsid=someAPS" rotate="45 0 0 combine" />
```

The single exception to the use of consistent encodings is for the 'linkuri' APS attribute, which is encoded as an element in the XCF, for reasons as explained.

See the WebCGM DOM data types section for complete details.

4.3.2 Conventions used

Most of the XCF elements may have Style Properties as XML attributes (Style Properties are defined and supported on the DOM WebCGMPicture and WebCGMAppStructure interfaces.) The following entity definition is used in the DTD snippets of the subsequent subsections, on those elements which support Style Properties at both the APS level and Picture level. (The background-color Style Property that applies only at the Picture level.)

```
<!ENTITY % styleProperties
                   "text-size            CDATA        #IMPLIED
                    fill-color           CDATA        #IMPLIED
                    intensity            CDATA        #IMPLIED
                    stroke-color         CDATA        #IMPLIED
                    stroke-weight        CDATA        #IMPLIED
                    text-color           CDATA        #IMPLIED
                    text-font            CDATA        #IMPLIED
                    raster-intensity     CDATA        #IMPLIED
                    stroke-type          CDATA        #IMPLIED
                    stroke-offset        CDATA        #IMPLIED
                    interior-style       CDATA        #IMPLIED
                    hatch-index          CDATA        #IMPLIED
                    pattern-index        CDATA        #IMPLIED
                    edge-visibility      CDATA        #IMPLIED
                    fill-offset          CDATA        #IMPLIED"
>
```

A single WebCGM XCF element must not contain both the `intensity` property and one or more of the overlapping properties `fill-color`, `stroke-color`, `text-color`. Overlapping properties may occur sequentially on different XCF elements, and then their processing is defined by their order of occurrence (see Style Properties description).

See "Data types and encoding" for more about Style Properties.

Some of the XCF elements may have geometric transformsas XML attributes (geometric transform Style Properties are defined and supported on the DOM WebCGMAppStructure interface.) The following entity definition is used in the DTD snippets of the subsequent subsections, on those elements which support geometric transform..

```
<!ENTITY % geometricTransform
                   "translate            CDATA        #IMPLIED
                    rotate               CDATA        #IMPLIED
                    scale                CDATA        #IMPLIED
                    setTransform         CDATA        #IMPLIED"
>
```

4.3.3 The 'webcgm' element

A WebCGM companion file (or any other CGM profile derived from the WebCGM profile) must have a 'webcgm' element as the root element. The 'webcgm' element corresponds to the Picture node in the WebCGM DOM tree (see for example the WebCGMPicture interface in the DOM).

```
<!ENTITY % webcgmEXT "" >
<!ENTITY % webcgmAttEXT "" >
<!ELEMENT webcgm ( (layer | grobject | para | subpara | bindById | bindByName %webcgmEXT;)* ) >
<!ATTLIST webcgm  id       ID    #IMPLIED
                  version  CDATA #FIXED '2.1'
                  filename CDATA #IMPLIED
                  xmlns    CDATA #FIXED "http://www.cgmopen.org/schema/webcgm/"
                  background-color  CDATA  #IMPLIED
                  pictureVisibility    ( on | off) #IMPLIED
                  %styleProperties;
                  %webcgmAttEXT;
>
```

Attribute definitions:

id="xml:id"
Standard XML attribute for assigning a unique identifier to an element. Refer to *Extensible Markup Language (XML) 1.0, 3rd Edition* [XML10].

version="CDATA"
Represents the version of the WebCGM specification. The value is 2.1 for this specification. Every conforming XCF must identify its version, either by including this attribute on the webcgm element, or by including a DOCTYPE pointing to this WebCGM XCF's DTD, or both (recommended). An industry-specific profile derived from this WebCGM XCF specification must not use this attribute to identify its version, and should define and require use of a namespace attribute to identify its profile version. For example, if ASD includes in its future version of S1000D "n.m" an XCF derived from WebCGM 2.1, the `webcgm` tag might look like this:

```
<webcgm version="2.1" asd:s1000d-version="n.m" xmlns:asd="http://example.org/asd/" ...>
```

filename="CDATA"
Represents the filename of the corresponding WebCGM file. 'filename' is a descriptive attribute.

xmlns[:prefix]="CDATA"
Standard XML attribute whose value defines the "resource-name" for identifying an XML namespace. Refer to Namespaces in XML. The 'xmlns' attribute without prefix identifies the (default) WebCGM namespace, and (with prefix) must be used to identify the foreign namespace(s) of any application-specific metadata that are used in the XCF instance. Note that the value given in the DTD snippet, and the attribute type (#FIXED), apply only to the unprefixed, default WebCGM namespace.

EXAMPLES:

```
<webcgm version="2.1" xmlns="http://www.cgmopen.org/schema/webcgm/"
                      xmlns:asd="http://example.org/asd/" ...>


<xcf:webcgm version="2.1" xmlns:asd="http://example.org/asd/"
                          ...>
```

In the first example, the WebCGM namespace is declared as the default namespace for the `webcgm` element and its contents, and content in the 'asd' namespace would use the "asd:" prefix. As the DTD shows, the second example is also valid because the namespace IRI defaults properly for the `xmlns` attribute without prefix. It is recommended, however, that the `xmlns` attribute declaring the default (WebCGM) namespace always be included.

background-color="CDATA"
background-color is a Style Property for setting the background color of the Picture (root node of WebCGM object tree).

pictureVisibility="on|off"
Defines the visibility for the picture that corresponds to the `webcgm` element of this XCF. This is similar to the 'visibility' APS Attribute that can be applied to metafile Application Structures, except it applies to the picture (on which the 'visibility' APS Attribute is not allowed by CGM rules.) The effect is the same as the invocation of the setPictureVisibility method on the WebCGMPicture interface of the DOM. .
styleProperties
the styleProperties entity collectively defines those Style Properties that apply at both the Picture level and the object/APS level.
webcgmEXT
the webcgmEXT entity is a mechanism for adding additional child content (i.e., metadata) on the root node.
webcgmAttEXT
the webcgmAttEXT entity is a mechanism for adding additional attributes (i.e., metadata) on the root node.

4.3.4 The 'layer' element

The 'layer' element of an XML companion file represents a CGM Application Structure of type 'layer'. The corresponding 'layer' is identifiable given its assigned 'apsid' attribute value.

```
<!ENTITY % layerEXT "EMPTY" >
<!ENTITY % layerAttEXT ""   >
<!ELEMENT layer %layerEXT;  >
<!ATTLIST layer apsid         ID           #REQUIRED
                layerdesc     CDATA        #IMPLIED
                visibility    ( on | off  | inherit) #IMPLIED
                interactivity ( on | off  | inherit) #IMPLIED
                %styleProperties;
                %geometricTransform;
                %layerAttEXT;
>
```

Attribute definitions:
apsid="xml:id"
The unique identifier of the Application Structure for the given WebCGM file.
layerdesc="CDATA"
Value of the 'layerdesc' Application Structure attribute for the associated APS.
visibility="on|off|inherit"
Value of the 'visibility' Application Structure attribute for the associated APS.
interactivity="on|off|inherit"
Value of the 'interactivity' Application Structure attribute for the associated APS.
styleProperties
the styleProperties entity collectively defines those Style Properties that apply at both the Picture level and the object/APS level.
geometricTransform
the geometricTransform entity collectively defines those geometric transforms that apply at the object/APS level.
layerEXT
the layerEXT entity is a mechanism for adding additional child content (i.e., metadata) on the 'layer'.
layerAttEXT
the layerAttEXT entity is a mechanism for adding additional attributes (i.e., metadata) on the 'layer'.

See also the 'layer' functional description in Section 3.

4.3.5 The 'grobject' element

The 'grobject' element of an XML companion file represents a CGM Application Structure of type 'grobject'. The corresponding grobject' is identifiable given its assigned 'apsid' attribute value.

```
<!ENTITY % grobjectEXT "" >
<!ENTITY % grobjectAttEXT "" >
<!ELEMENT grobject ( linkuri %grobjectEXT; )* >
<!ATTLIST grobject  apsid         ID           #REQUIRED
                    screentip     CDATA        #IMPLIED
                    region        CDATA        #IMPLIED
                    viewcontext   CDATA        #IMPLIED
                    visibility    ( on | off | inherit) #IMPLIED
                    interactivity ( on | off | inherit) #IMPLIED
                    %styleProperties;
                    %geometricTransform;
                    %grobjectAttEXT;
>
```

Attribute definitions:
apsid="xml:id"
The unique identifier of the Application Structure for the given WebCGM file.
screentip="CDATA"
Value of the 'screentip' Application Structure attribute for the associated APS.
region="CDATA"
Value of the 'region' Application Structure attribute for the associated APS.
viewcontext="CDATA"
Value of the 'viewcontext' Application Structure attribute for the associated APS.
visibility="on|off|inherit"
Value of the 'visibility' Application Structure attribute for the associated APS.
interactivity="on|off|inherit"
Value of the 'interactivity' Application Structure attribute for the associated APS.
styleProperties
the styleProperties entity collectively defines those Style Properties that apply at both the Picture level and the object/APS level.
geometricTransform
the geometricTransform entity collectively defines those geometric transforms that apply at the object/APS level.
grobjectEXT
the grobjectEXT entity is a mechanism for adding additional child content (i.e., metadata) on the 'grobject'.
grobjectAttEXT
the grobjectAttEXT entity is a mechanism for adding additional attributes (i.e., metadata) on the 'grobject'.

See also the 'grobject' functional description in Section 3.

4.3.6 The 'para' element

The 'para' element of an XML companion file represents a CGM Application Structure of type 'para'. The corresponding 'para' is identifiable given its assigned 'apsid' attribute value.

```
<!ENTITY % paraEXT "" >
<!ENTITY % paraAttEXT "" >
<!ELEMENT para ( linkuri %paraEXT; )* >
<!ATTLIST para apsid         ID           #REQUIRED
               screentip     CDATA        #IMPLIED
               region        CDATA        #IMPLIED
               viewcontext   CDATA        #IMPLIED
               visibility    ( on | off | inherit) #IMPLIED
               interactivity ( on | off | inherit) #IMPLIED
               %styleProperties;
               %geometricTransform;
               %paraAttEXT;
>
```

Attribute definitions:
apsid="xml:id"
The unique identifier of the Application Structure for the given WebCGM file.
screentip="CDATA"
Value of the 'screentip' Application Structure attribute for the associated APS.
region="CDATA"
Value of the 'region' Application Structure attribute for the associated APS.
viewcontext="CDATA"
Value of the 'viewcontext' Application Structure attribute for the associated APS.
visibility="on|off|inherit"
Value of the 'visibility' Application Structure attribute for the associated APS.
interactivity="on|off|inherit"
Value of the 'interactivity' Application Structure attribute for the associated APS.
styleProperties
the styleProperties entity collectively defines those Style Properties that apply at both the Picture level and the object/APS level.
geometricTransform
the geometricTransform entity collectively defines those geometric transformsthat apply at the object/APS level.
paraEXT
the paraEXT entity is a mechanism for adding additional child content (i.e., metadata) on the 'para'.
paraAttEXT
the paraAttEXT entity is a mechanism for adding additional attributes (i.e., metadata) on the 'para'.

See also the 'para' functional description in Section 3.

4.3.7 The 'subpara' element

The 'subpara' element of an XML companion file represents a CGM Application Structure of type 'subpara'. The corresponding 'subpara' is identifiable given its assigned 'apsid' attribute value.

```
<!ENTITY % subparaEXT "" >
<!ENTITY % subparaAttEXT "" >
<!ELEMENT subpara ( linkuri %subparaEXT; )* >
<!ATTLIST subpara apsid         ID           #REQUIRED
                  screentip     CDATA        #IMPLIED
                  region        CDATA        #IMPLIED
                  viewcontext   CDATA        #IMPLIED
                  visibility    ( on | off | inherit) #IMPLIED
                  interactivity ( on | off | inherit) #IMPLIED
                  %styleProperties;
                  %geometricTransform;
                  %subparaAttEXT;
>
```

Attribute definitions:
apsid="xml:id"
The unique identifier of a the Application Structure for the given WebCGM file.
screentip="CDATA"
Value of the 'screentip' Application Structure attribute for the associated APS.
region="CDATA"
Value of the 'region' Application Structure attribute for the associated APS.
viewcontext="CDATA"
Value of the 'viewcontext' Application Structure attribute for the associated APS.
visibility="on|off|inherit"
Value of the 'visibility' Application Structure attribute for the associated APS.
interactivity="on|off|inherit"
Value of the 'interactivity' Application Structure attribute for the associated APS.
styleProperties
the styleProperties entity collectively defines those Style Properties that apply at both the Picture level and the object/APS level.
geometricTransform
the geometricTransform entity collectively defines those geometric transforms that apply at the object/APS level.
subparaEXT
the subparaEXT entity is a mechanism for adding additional child content (i.e., metadata) on the 'subpara'.
subparaAttEXT
the subparaAttEXT entity is a mechanism for adding additional attributes (i.e., metadata) on the 'subpara'.

See also the 'subpara' functional description in Section 3.

4.3.8 The 'linkuri' element

A 'linkuri' element of an XML companion file represents a WebCGM 'linkuri' Application Structure attribute. Contrary to other attributes, the 'linkuri' attribute is expressed as an element in the XML companion file. The corresponding Application Structure of this 'linkuri' is its parent element.

```
<!ENTITY % linkuriEXT "" >
<!ENTITY % linkuriAttEXT "" >
<!ELEMENT linkuri %linkuriEXT; >
<!ATTLIST linkuri uri      CDATA #REQUIRED
                  behavior CDATA #IMPLIED
                  desc     CDATA #IMPLIED
                  %linkuriAttEXT;
>
```

Attribute definitions:
uri="CDATA"
The IRI of this 'linkuri' attribute. See section *Basic Data Types* for more information.
behavior="CDATA"
The behavior of this 'linkuri' attribute. See section *Basic Data Types* for more information.
desc="CDATA"
The title or description of this 'linkuri' attribute. See section *Basic Data Types* for more information.
linkuriEXT
the linkuriEXT entity is a mechanism for adding additional child content (i.e., metadata) on the 'linkuri'.
linkuriAttEXT
the linkuriAttEXT entity is a mechanism for adding additional attributes (i.e., metadata) on the 'linkuri'.

See also the 'linkuri' functional description in Section 3.

4.3.9 The 'bindByName' element

A 'bindByName' element of an XML companion file is intended to correspond to one or more Application Structure in a CGM file. The common link between those Application Structures is that their 'name' or 'layername' attribute value corresponds to 'apstargetname'. See section *Relationship with XML companion file* for more information on the rules of mapping 'bindByName' attributes to WebCGM Application Structures.

```
<!ENTITY % bindByNameEXT "" >
<!ENTITY % bindByNameAttEXT "" >
<!ELEMENT bindByName ( linkuri %bindByNameEXT; )* >
<!ATTLIST bindByName apstargetname CDATA        #REQUIRED
                     screentip     CDATA        #IMPLIED
                     region        CDATA        #IMPLIED
                     viewcontext   CDATA        #IMPLIED
                     layerdesc     CDATA        #IMPLIED
                     visibility    ( on | off | inherit) #IMPLIED
                     interactivity ( on | off | inherit) #IMPLIED
                     %styleProperties;
                     %bindByNameAttEXT;
>
```

Attribute definitions:
apstargetname="CDATA"
Name used to identify the corresponding Application Structure(s) for a given WebCGM file.
screentip="CDATA"
Value of the 'screentip' Application Structure attribute for the associated APS.
region="CDATA"
Value of the 'region' Application Structure attribute for the associated APS.
viewcontext="CDATA"
Value of the 'viewcontext' Application Structure attribute for the associated APS.
layerdesc="CDATA"
Value of the 'layerdesc' Application Structure attribute for the associated APS.
visibility="on|off|inherit"
Value of the 'visibility' Application Structure attribute for the associated APS.
interactivity="on|off|inherit"
Value of the 'interactivity' Application Structure attribute for the associated APS.
styleProperties
the styleProperties entity collectively defines those Style Properties that apply at both the Picture level and the object/APS level.
bindByNameEXT
the bindByNameEXT entity is a mechanism for adding additional child content (i.e., metadata) on the APS.
bindByNameAttEXT

the bindByNameAttEXT entity is a mechanism for adding additional attributes (i.e., metadata) on the APS.

4.3.10 The 'bindById' element

The 'bindById' element of an XML companion file represents a CGM Application Structure one of the types: layer, grobject, para, subpara. APS of type 'grnode' are valid in a 'bindById' element. The corresponding object is identifiable given its assigned 'apsid' attribute value. See section *Relationship with XML companion file* for more information on the rules of mapping 'bindById' attributes to WebCGM Application Structures.

```
<!ENTITY % bindByIdEXT "" >
<!ENTITY % bindByIdAttEXT "" >
<!ELEMENT bindById ( linkuri %bindByIdEXT; )* >
<!ATTLIST bindById apsid         ID           #REQUIRED
                   screentip     CDATA        #IMPLIED
                   region        CDATA        #IMPLIED
                   viewcontext   CDATA        #IMPLIED
                   layerdesc     CDATA        #IMPLIED
                   visibility    ( on | off | inherit) #IMPLIED
                   interactivity ( on | off | inherit) #IMPLIED
                   %styleProperties;
                   %bindByIdAttEXT;
>
```

Attribute definitions:
apsid="xml:id"
The unique identifier of the Application Structure for the given WebCGM file.
screentip="CDATA"
Value of the 'screentip' Application Structure attribute for the associated APS.
region="CDATA"
Value of the 'region' Application Structure attribute for the associated APS.
viewcontext="CDATA"
Value of the 'viewcontext' Application Structure attribute for the associated APS.
layerdesc="CDATA"
Value of the 'layerdesc' Application Structure attribute for the associated APS.
visibility="on|off|inherit"
Value of the 'visibility' Application Structure attribute for the associated APS.
interactivity="on|off|inherit"
Value of the 'interactivity' Application Structure attribute for the associated APS.
styleProperties
the styleProperties entity collectively defines those Style Properties that apply at both the Picture level and the object/APS level.
bindByIdEXT
the bindByIdEXT entity is a mechanism for adding additional child content (i.e., metadata) on the APS.
bindByIdAttEXT
the bindByIdAttEXT entity is a mechanism for adding additional attributes (i.e., metadata) on the APS.

4.4 The complete XCF DTD

This section is normative.

The complete WebCGM XML Companion File (XCF) DTD follows.

```
<?xml version="1.0" encoding="UTF-8"?>>
<!-- ============================================================= -->
<!-- This is the WebCGM XML Companion File DTD for use with        -->
```

```
<!-- WebCGM 2.1                                                          -->

<!-- ================================================================== -->
<!-- Original issue: March 2008                                         -->
<!--                                                                    -->
<!-- Revision history:                                                  -->
<!--        Original.                                                   -->
<!--                                                                    -->
<!-- ================================================================== -->
<!--                                                                    -->
<!-- ================================================================== -->

<!-- Application specific entities                                      -->
<!-- Application groups define application specific attributes here     -->
<!-- and define the stubs for application specific elements that        -->
<!-- will be defined later in the DTD                                   -->
<!--                                                                    -->

<!ENTITY % webcgmEXT ""                                                  >
<!ENTITY % webcgmAttEXT ""                                               >
<!ENTITY % layerEXT "EMPTY"                                              >

<!ENTITY % layerAttEXT ""                                                >
<!ENTITY % grobjectEXT ""                                                >
<!ENTITY % grobjectAttEXT ""                                             >
<!ENTITY % paraEXT ""                                                    >
<!ENTITY % paraAttEXT ""                                                 >
<!ENTITY % subparaEXT ""                                                 >
<!ENTITY % subparaAttEXT ""                                              >
<!ENTITY % linkuriEXT "EMPTY"                                            >

<!ENTITY % linkuriAttEXT ""                                              >
<!ENTITY % bindByIdEXT ""                                                >
<!ENTITY % bindByIdAttEXT ""                                             >
<!ENTITY % bindByNameEXT ""                                              >
<!ENTITY % bindByNameAttEXT ""                                           >
<!ENTITY % styleProperties
                   "text-size           CDATA        #IMPLIED
                    fill-color          CDATA        #IMPLIED
                    intensity           CDATA        #IMPLIED
                    stroke-color        CDATA        #IMPLIED
                    stroke-weight       CDATA        #IMPLIED
                    text-color          CDATA        #IMPLIED
                    text-font           CDATA        #IMPLIED
                    raster-intensity    CDATA        #IMPLIED
                    stroke-type         CDATA        #IMPLIED
                    stroke-offset       CDATA        #IMPLIED
                    interior-style      CDATA        #IMPLIED
                    hatch-index         CDATA        #IMPLIED
                    pattern-index       CDATA        #IMPLIED
                    edge-visibility     CDATA        #IMPLIED
                    fill-offset         CDATA        #IMPLIE          >
<!ENTITY % geometricTransform
                   "translate           CDATA        #IMPLIED
                    rotate              CDATA        #IMPLIED
                    scale               CDATA        #IMPLIED
                    setTransform        CDATA        #IMPLIED"        >
<!--                                                                   -->
<!ELEMENT webcgm ( (layer | grobject | para | subpara |
                    bindById | bindByName %webcgmEXT;)* )             >
```

```
<!ATTLIST webcgm id       ID    #IMPLIED
                 version  CDATA #FIXED '2.1'
                 filename CDATA #IMPLIED
                 background-color CDATA #IMPLIED
                 pictureVisibility ( on | off ) #IMPLIED
                 xmlns    CDATA #FIXED "http://www.cgmopen.org/schema/webcgm/"
                 %styleProperties;
                 %webcgmAttEXT;                                          >

<!ELEMENT layer %layerEXT;                                               >
<!ATTLIST layer apsid         ID           #REQUIRED
                layerdesc     CDATA        #IMPLIED
                visibility    ( on | off | inherit) #IMPLIED
                interactivity ( on | off | inherit) #IMPLIED
                %styleProperties;
                %geometricTransform;
                %layerAttEXT;                                            >

<!ELEMENT grobject ( linkuri %grobjectEXT; )*                            >
<!ATTLIST grobject apsid         ID           #REQUIRED
                   screentip     CDATA        #IMPLIED
                   region        CDATA        #IMPLIED
                   viewcontext   CDATA        #IMPLIED
                   visibility    ( on | off | inherit) #IMPLIED
                   interactivity ( on | off | inherit) #IMPLIED
                   %styleProperties;
                   %geometricTransform;
                   %grobjectAttEXT;                                      >

<!ELEMENT linkuri %linkuriEXT;                                           >
<!ATTLIST linkuri uri      CDATA #REQUIRED
                  behavior CDATA #IMPLIED
                  desc     CDATA #IMPLIED
                  %linkuriAttEXT;                                        >

<!ELEMENT para ( linkuri %paraEXT; )*                                    >
<!ATTLIST para apsid       ID           #REQUIRED
               screentip     CDATA        #IMPLIED
               region        CDATA        #IMPLIED
               viewcontext   CDATA        #IMPLIED
               visibility    ( on | off | inherit) #IMPLIED
               interactivity ( on | off | inherit) #IMPLIED
               %styleProperties;
               %geometricTransform;
               %paraAttEXT;                                              >

<!ELEMENT subpara ( linkuri %subparaEXT; )*                              >
<!ATTLIST subpara apsid         ID           #REQUIRED
                  screentip     CDATA        #IMPLIED
                  region        CDATA        #IMPLIED
                  viewcontext   CDATA        #IMPLIED
                  visibility    ( on | off | inherit) #IMPLIED
                  interactivity ( on | off | inherit) #IMPLIED
                  %styleProperties;
                  %geometricTransform;
                  %subparaAttEXT;                                        >

<!ELEMENT bindById ( linkuri %bindByIdEXT; )*                            >
<!ATTLIST bindById apsid         ID           #REQUIRED
```

```
                  screentip     CDATA        #IMPLIED
                  layerdesc     CDATA        #IMPLIED
                  region        CDATA        #IMPLIED
                  viewcontext   CDATA        #IMPLIED
                  visibility    ( on | off | inherit) #IMPLIED
                  interactivity ( on | off | inherit) #IMPLIED
                  %styleProperties;
                  %geometricTransform;
                  %bindByIdAttEXT;                                    >

<!ELEMENT bindByName ( linkuri %bindByNameEXT; )*                     >

<!ATTLIST bindByName apstargetname CDATA        #REQUIRED
                     screentip     CDATA        #IMPLIED
                     layerdesc     CDATA        #IMPLIED
                     region        CDATA        #IMPLIED
                     viewcontext   CDATA        #IMPLIED
                     visibility    ( on | off | inherit) #IMPLIED
                     interactivity ( on | off | inherit) #IMPLIED
                     %styleProperties;
                     %geometricTransform;
                     %bindByNameAttEXT;                                 >
<!--                                                                   -->
<!-- Define content models for application specific elements          -->
<!--                                                                   -->
```

Back to top of chapter

---

5. WebCGM Document Object Model (DOM)

This chapter and its sections are normative, unless otherwise indicated.

Contents

5.1 Overview

This section is informative (non-normative).

This chapter defines a set of objects and interfaces for accessing and manipulating WebCGM documents. The functionality specified in this section enables script writers to manipulate WebCGM documents and access information found in standard WebCGM XML companion files. The WebCGM DOM API focuses its methods on: tree traversal, style changes, and providing access to metadata.

5.2 Relationship with XML DOM

This section is informative (non-normative).

Although inspired by the XML DOM specifications, the WebCGM DOM remains oriented towards WebCGM specific functionality. Since WebCGM uses a tree structure to group graphical primitives, it was therefore appropriate, to use a set of interfaces similar to the XML DOM Node, Element and Document interfaces. However, since WebCGM is expressed in a non-XML syntax, several changes had to be made to commonly known interfaces and methods in order to improve the user experience of WebCGM script writers.

The WebCGM DOM could almost be perceived as a 'readonly' DOM. Some interface methods allow users to change the visual appearance of Application Structures, but unlike the XML DOM specification, it does not allow for removal or insertion of WebCGMNodes into the object model. This constitute a significant difference between the specifications.

While WebCGM 1.0 offered interactivity support via hyperlinking and highlighting, the WebCGM 2.0 DOM took it to the next level. WebCGM 2.1 further enhances the DOM. The WebCGM DOM borrows concepts from the DOM3 Events specification, and introduces the concept of EventListeners and mouse Events in order to meet the requirements of WebCGM users.

5.3 Relationship with XML companion file

The WebCGM DOM is designed to provide access to XML metadata found in XML Companion Files. Practice has shown that some CGM illustrations are easier to maintain if some of the non graphical information remains outside the illustration. An example of such information could be; language sensitive screentips. The WebCGM DOM can then be used to 'apply' the information from the XML companion file to the WebCGM document (see Example 5.3) . For more information on XML companion file syntax, please refer to ***Chapter 4, WebCGM XML Companion File***.

Another benefit of the XML Companion File is to carry application specific data (or metadata) concerning a WebCGM illustration (see Example 4.2). This information is expressed using namespace attributes and elements in the XML Companion File. The WebCGM DOM provides a method for loading the XML metadata into the user agent's object model. Using the WebCGM DOM, a user can gain access to the metadata. Here is a detailed example to better illustrate the concept.

Example 5.1a: *This WebCGM document (expressed in clearText encoding) will be updated by an XML companion file.*

```
BEGMF 'example.cgm';
...
BEGPIC 'Picture 1';
...
BEGAPS 'L1' 'layer' STLIST;
  APSATTR 'layername' "14 1 'Standard layer'";
  BEGAPSBODY;
  BEGAPS 'G1' 'grobject' STLIST;
    BEGAPSBODY;
    LINE 210,265 210,200 300,200;
    LINE 300,200 300,265 210,265;
  ENDAPS;
ENDAPS;
...
ENDPIC;
...
ENDMF;
```

The in-memory tree representation of this illustration should be similar to the illustration found below. The metafile contains a picture, the picture contains a child node Application Structure of type layer, and the layer contains a child node Application Structure of type grobject, as illustrated in Figure 7.

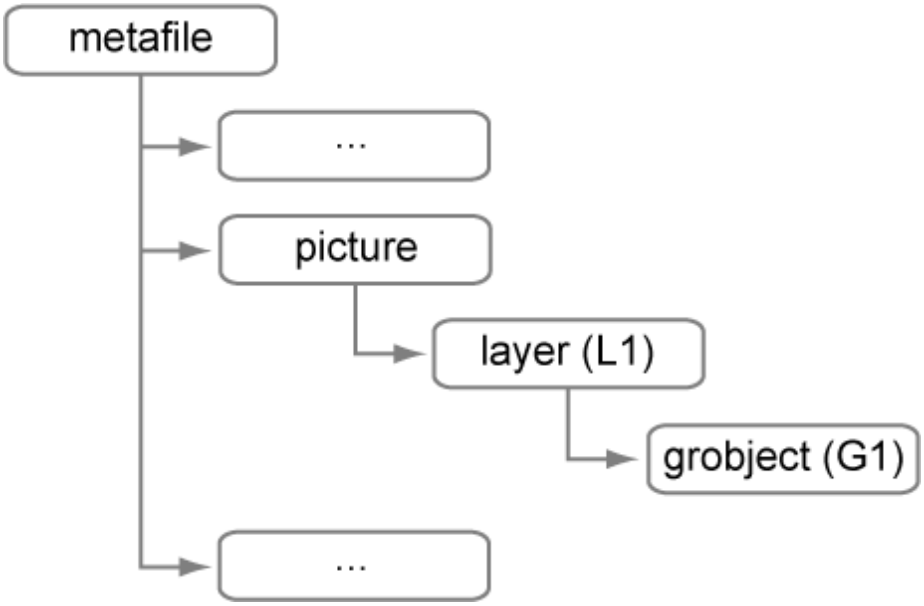

Figure 7. Original tree structure

Example 5.1b: *XML companion file to be 'applied' on example.cgm of example 5.1a.*

```
<webcgm id="example" xmlns:wiring="http://www.example.org">
  <grobject apsid="G1" screentip="A new screentip">
    <wiring:data wire-bundle="E132-NAV"/>
  </grobject>
</webcgm>
```

The WebCGM DOM provides methods for 'applying' an XML Companion File, like the one shown in example 5.1b, to a picture in a WebCGM document. A conforming user agent is expected to load and parse the XML Companion File and possibly 'apply' updates from the XML Companion File to the user agent's object model. A user may want to apply a companion file for the following reasons:

i) To replace standard Application Structure Attribute values present in the WebCGM instance with new values from the XML Companion File.

ii) To supply standard Application Structure Attribute values to Application Structures which do not contain attribute values with values from the XML companion file.

iii) To transiently modify the Style Properties (stoke-color, text-size, etc) with which an object (APS or picture) is displayed.

iv) To add XML metadata to the user agent's object model to be retrieved at a later stage using WebCGM DOM APIs.

Once the user agent has loaded the XML Companion File into its memory model, the tree should resemble this:

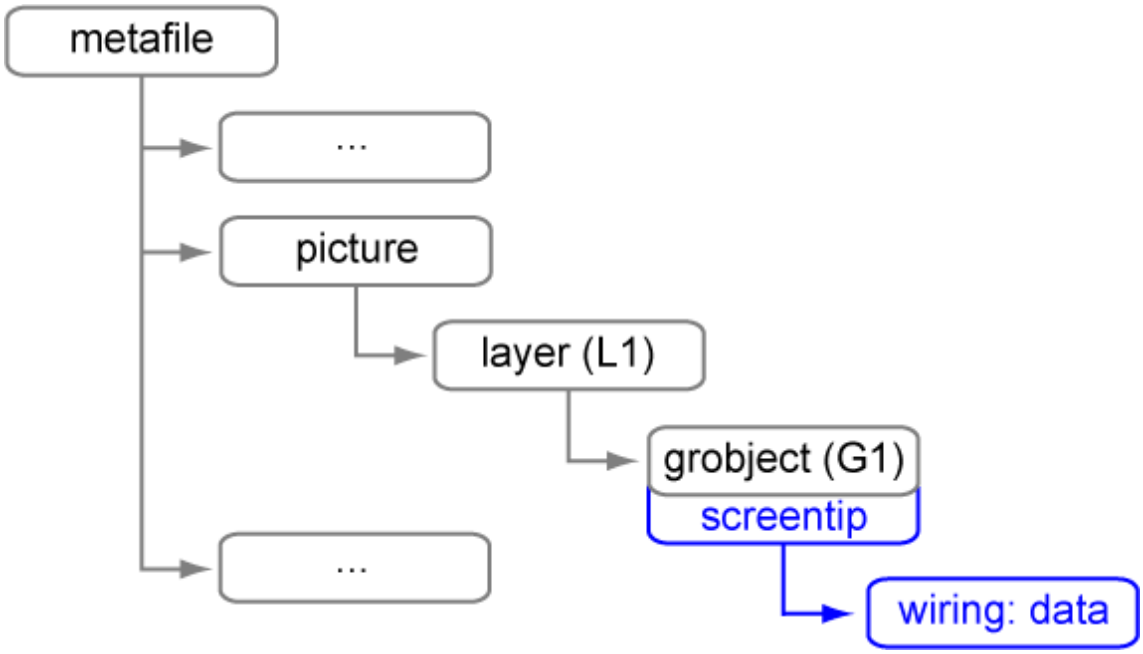

Figure 8. New tree structure

The overall set of rules that a user agent must follow when applying an XML Companion File is as follows:

1. Verify that root element is <webcgm>, else stop further processing and throw FILE_INVALID_ERR exception.
2. Process unknown attributes if any on root element, see below about **processing namespace attributes**.
3. Process all child elements using a depth-first algorithm.

More specific rules for **processing namespace attributes** are:

1. If the target APS is not present in the CGM file, all attributes of the current element are ignored.
2. If the attribute is not part of the base WebCGM DTD, it must be an extended namespace attribute, else the attribute is ignored.
3. An attribute that already exists on the corresponding APS must be updated with the new attribute value.
4. In the case where an attribute with the same local name and namespace IRI is already present on the APS, its prefix is changed to be the prefix part of the qualifiedName, and its value is changed to be the attribute value. If the attribute does not exist on the APS, the namespace attribute is appended onto the APS.

More specific rules for **processing child elements** are:

1. The target APS (the parent element) must be present in the CGM file, if that is not the case, all child elements of the current element are ignored.
2. The target APS must not be of type 'grnode'. Type 'grnode' elements are not accessible via XCF. Nor are they accessible via most DOM calls, with the principal exception of the WebCGMNode interface (firstChild, nextSibling, etc).
3. If the element is not part of the base WebCGM DTD, it must be an extended namespace element. Namespace elements and their attributes are appended at the end of the target's list of child elements.
4. Elements that are defined in the WebCGM DTD are processed as follows:
    - namespace attributes are processed as specified above.
    - attributes relevant to this element, are updated on the APS.
    - other attributes are ignored.
5. If the element is a <linkuri>, the following rules apply:
    - If one or more 'linkuri' attribute(s) already exist on the parent element, they are all deleted and replaced with the corresponding set of 'linkuri's from the XCF. It is not possible to add links from the XCF to already existing links.
    - The attributes of each <linkuri> element are combined to create a single WebCGM 'linkuri' APS attribute (as defined in WebCGM 1.0) on the parent element.
6. If the element is a <bindById>, only namespace attributes and attributes relevant to the target APS type are added to the target (i.e., if the <bindById> has a 'screentip' attribute and the target APS is of type 'layer', the 'screentip' attribute will be ignored).
7. If the element is a <bindByName>, the user agent has to find all Application Structures that have a matching 'name' or 'layername' attribute. All found Application Structures are then subject to new attribute values (refer to the <bindById> description above).

5.4 Inheritance of APS Attributes and of Style Properties

This section describes how APS Attributes are inherited in a WebCGM structure tree. It also describes how Style Properties are inherited, which is similar but differs in a few key details. The inheritance models are based closely on the inheritance model of CSS 2.0. Some details have been adapted to the particulars of the WebCGM format. This chapter is the normative reference for inheritance of APS Attributes and of Style Properties in WebCGM.

5.4.1 Specified, computed, and actual values of Style Properties

WebCGM user agents are required to support the inheritance model defined in this section for eligible Style Properties. Once a user agent has loaded a document and constructed a document tree, it must assign, for every Application Structure in the tree, a value to every Style Property.

Very similar to the CSS model, the final value of a Style Properties is the result of a four-step calculation: the value is determined through specification (the "Specified Value"), then resolved into a value that may be used for inheritance (the "Computed Value"), then converted into an absolute value if necessary (the "Used Value"), and finally transformed according to the limitations of the local environment (the "Actual Value").

5.4.1.1 Specified Values of Style Properties

User agents must first assign a Specified Value to each Style Property based on the following mechanisms (in order of precedence):

1. If the Style Property is assigned a value (via a DOM call, or via an XCF), use it.
2. Otherwise, if the Style Property is inherited (i.e., its definition includes "Inherited:yes") and the Application Structure is not the root of the document tree, use the Computed Value of the parent Application Structure.
3. Otherwise use the Style Property's Initial Value. The Initial Value of each Property is indicated in the Style Property's definition.

Note: In the context of WebCGM inheritance, the root of the document tree is the Picture node.

5.4.1.2 Computed Values of Style Properties

Specified Values are resolved to Computed Values after the document tree is created.

- When the Specified Value is not 'inherit', the Computed Value of a Style Property is determined as specified by the Computed Value line in the definition of the Property.
- When the Specified Value is 'inherit', it must be replaced for the Computed Value as defined below in the section on inheritance.

The Computed Value exists even when the property doesn't apply, as defined by the 'Applies To' line.

5.4.1.3 Used Values of Style Properties

In the CSS2 model from which the WebCGM model is derived, Computed Values can be relative to each other; for example a width could be set as a percentage, which is dependent on the containing block's width. The Used Value is the result of taking the Computed Value and resolving these dependencies into a final absolute value used for the actual display. In this version of WebCGM, there are no examples where Used Value differs from the Computed Value. This may change in a future version of the specification.

5.4.1.4 Actual Values of Style Properties

A Used Value is in principle the value used for rendering, but a user agent may not be able to make use of the value in a given environment. For example, a user agent may only be able to render borders with integer pixel widths and may therefore have to approximate the computed width, or the user agent may be forced to use only black and white shades instead of full colour. The Actual Value is the used value after any approximations have been applied.

5.4.2 Specified, computed, and actual values of Application Structure Attributes

WebCGM user agents are required to support the inheritance model of Application Structure (APS) Attributes defined in this section. Once a user agent has loaded a document and constructed a document tree, it must resolve, for every Application Structure in the tree, if an Attribute has a value (i.e., no value is possible for some attributes).

Very similar to the CSS model, the final value of an APS Attribute is the result of a four-step calculation: the value is determined through specification (the "Specified Value"), then resolved into a value that may be used for inheritance (the "Computed Value"), then converted into an absolute value if necessary (the "Used Value"), and finally transformed according to the limitations of the local environment (the "Actual Value").

5.4.2.1 Specified Values of Application Structure Attributes

User agents must first assign a Specified Value to each APS Attribute based on the following mechanisms (in order of precedence):

1. If the Attribute is assigned a value in the CGM document or assigned a new value via a DOM call or an XCF, use it.
2. Otherwise, if the Attribute is inherited (i.e., its definition includes "Inherited:yes") and the Application Structure is not the root of the document tree, use the Computed Value of the parent Application Structure.
3. Otherwise use the Attribute's Initial Value. The Initial Value of each Attribute is indicated in the Attribute's definition.

In the context of WebCGM inheritance, the root of the document tree is the Picture node. For the purposes of this inheritance model, the Picture root node is treated as if it were an Application Structure.

5.4.2.2 Computed Values of Application Structure Attributes

In this specification, Computed Values of Application Structure Attributes, with the exception of the 'inherit' value are identical to the Specified Values. When the Specified Value is 'inherit', it must be replaced for the Computed Value as defined below in the section on inheritance. The Computed Value exists even when the Attribute doesn't apply (as defined by the 'Applies To' line in the Attribute's definition).

5.4.2.3 Used Values of Application Structure Attributes

In the CSS2 model from which the WebCGM model is derived, Computed Values can be relative to each other; for example a width could be set as a percentage, which is dependent on the containing block's width. The Used Value is the result of taking the Computed Value and resolving these dependencies into a final absolute value used for the actual display. In this version of WebCGM, there are no examples where Used Value differs from the Computed Value. This may change in a future version of the specification.

5.4.2.4 Actual Values of Application Structure Attributes

A Used Value is in principle the value used for rendering, but a user agent may not be able to make use of the value in a given environment. For example, a user agent may only be able to render borders with integer pixel widths and may therefore have to approximate the computed width, or the user agent may be forced to use only black and white shades instead of full colour. The Actual Value is the Used Value after any approximations have been applied. In this version of WebCGM, there are no examples where Actual Value differs from the Used Value. This may change in a future version of the specification.

5.4.3 Inheritance

Some values are inherited by the children of an Application Structure in the document tree, as described above. Each Style Property and Application Structure Attribute defines whether it is inherited or not. As a general rule, when inheritance occurs, Application Structures inherit Computed values of Style Properties and Application Structure Attributes (unless implicitly stated in the Property or Attribute definition).

5.4.3.1 The 'inherit' value

Application Structure Attributes and Style Properties may also have a Specified Value of 'inherit'; which means that, for a given Application Structure, the Property or Attribute takes the same Computed value of the Application Structure's parent. The 'inherit' value can be used to strengthen inherited values, and it can also be used on Style Properties that are not normally inherited. There are no examples of the latter in this version of Web CGM.

5.5 Basic data types

WebCGM consists of three components where data type definitions need to be considered: metafile instances, DOM, and XCF.

5.5.1 Metafile data types

WebCGM instances are binary files. Metafile data types and encodings are fully defined in the CGM:1999 standard, together with chapters 3 and 6 of this WebCGM specification.

5.5.2 DOM data types

5.5.2.1 IDL types and binding types

Each interface of this DOM definition is normatively specified by a section of IDL code. The IDL uses basic data types such as unsigned short, boolean, long, etc. DOM applications are written in some programming language binding of the IDL, such as ECMAScript or Java. The only normatively specified binding for WebCGM DOM is the ECMAScript binding.

The ECMAScript binding unambiguously associates ECMAScript language data types with the IDL data types, which provides all the data type information needed to write WebCGM DOM applications in ECMAscript.

Null return value. WebCGM DOM attributes and method return values of type WebCGMNode and WebCGMNodeList sometimes have to represent the case of no data, i.e., zero nodes. In the DOM functional specification of this chapter, the term **null** is uniformly used to used to represent this case. In DOM bindings such as the ECMAScript binding the **null** return value maps naturally to an ECMAScript `null` reserved keyword. (For example, `myNode.childnodes==null` evaluates to true, and `myPicture.getAppStructureById()==null` evaluates to true.)

5.5.2.2 The WebCGMString type

One heavily used data type in the IDL definition is the WebCGMString, and some of the DOM interfaces do specify substructure or sub-types for some WebCGMString attributes and parameters.

WebCGMString

A WebCGMString is a sequence of 16-bit units in WebCGM DOM.

IDL Definition

```
valuetype WebCGMString sequence<unsigned short>;
```

In WebCGM DOM, like XML DOM Level 3, the UTF-16 encoding was chosen because of its widespread industry practice. For ECMAScript and Java, WebCGMString is bound to the String type because both languages also use UTF-16 as their encoding. The WebCGM DOM has many interfaces that imply string matching. For XML, string comparisons are case-sensitive and performed with a binary comparison of the 16-bit units of the WebCGMStrings.

Empty string. WebCGM DOM attributes and method return values of type WebCGMString sometimes have to represent a string that has no data, i.e., zero characters. In the DOM functional specification of this chapter, the term **empty string** is uniformly used to used to represent this case. In DOM bindings such as the ECMAScript binding, the WebCGM empty string maps naturally to an ECMAScript string of length zero, i.e., zero characters. (For example, `myEmptyString.length==0` evaluates to true, and `myEmptyString==""` evaluates to true.)

5.5.2.3 WebCGMString sub-types

The WebCGM DOM has a number of WebCGMString attributes or parameters that in fact encode other data — such as numbers, colors, sub-strings, etc — into the string format. For the purposes of this specification, we define the following rules for how WebCGMString **sub-types** are encoded and represented within WebCGMString objects.

Number

A real number value encoded in a WebCGMString. The representation of the number can be either **decimal notation** or **scientific notation**. Decimal notation consists of either an integer, or an optional sign character followed by zero or more digits followed by a dot (.) followed by one or more digits. Scientific notation consists of a decimal-notation representation immediately followed by the letter "e" or "E" immediately followed by an integer.

Percent

A percent, encoded in a WebCGMString, is a **number** followed by a percent-sign character, "%".

Color

A color, encoded in a WebCGMString, is a "#", followed by exactly six hex digits, [0-9a-fA-F]. The first two digits represent the red component, the second two represent the green component, and the last two represent the blue component. Conceptually: #RRGGBB. Examples: #FF0000 is full red, #e1e1e1 is the gray background of the IDL definitions in this chapter, #00FFFF is full cyan, etc.

List-of-number

The following EBNF defines list-of-number (with number as defined above):

```
list-of-number  ::= number | number Wsp list-of-number
Wsp             ::= (#x20 | #x9 | #xD | #xA)+
```

(Note: Wsp matches the"Whitespace" definition of XML 1.0 [XML10].)

Delimited String (list-of-string)

Some WebCGMString attributes may encode multiple substrings, e.g., the APS Attributes, 'name' and 'linkuri'. For historical reasons, this is known as Delimited String sub-type (although functionally it is a "list-of-string").

A Delimited String conforms to the following notation:

```
DelimitedString ::= ListOfX | ListOfXX
ListX           ::= '"'Name'"' | '"'Name'"' Wsp ListX
ListXX          ::= "'"Name"'" | "'"Name"'" Wsp ListXX
Wsp             ::= (#x20 | #x9 | #xD | #xA)+
Name            ::= (ValidChar)*
```

The definition of ValidChar depends on the particular WebCGM entity that is being encoded. For example, in the APS Attributes table for DOM access (in section 5.7.6, Interface WebCGMAppStructure), the valid characters for each of the APS Attributes that are encoded by Delimited String are determined by the WebCGM datatype of the particular APS Attribute (linked from the table), in combination with the Character Repertoire rules of section 3.1.1.3.

In the case of the 'linkuri' APS attribute, the value always contains 3 * n strings, n representing the number of 'linkuri' attributes specified on the Application Structure. When meaningful values are not supplied for some of the components, the components must be represented by an empty string. The restriction of 3 * n strings simplifies scripts aimed at manipulating Delimited Strings.

> **Example**: to set a 'region' APS Attribute that consisted of two subregions:
> *setAppStructureAttr("region", ""1 0 0 100 100' '1 25 25 75 75"")*
>
> **Example**: a multilink consisting of two links could be represented with the following delimited string:
> *'http://www.w3.org/' 'W3C' '_blank' 'http://www.cgmopen.org/' 'CGMOpen' '_self'.*

A Delimited String is a list of wsp-separated substrings. If the Delimited String only contains a single substring, then it is coded as a simple string.

Example: to set a 'region' APS Attribute that consists of a single subregion:
setAppStructureAttr("region", "1 0 0 100 100")

Note. This Delimited String syntax, when combined with handling of string parameters in languages such as ECMAScript, imposes some constraints on the content of Delimited String substrings. In particular, it would not be possible to have a substring that contained both a QUOTE character and an APOSTROPHE character.

5.5.3 XCF data types

The XML Companion File (XCF) provides access to many of the APS Attributes and Style Properties that are accessible via the WebCGM DOM. APS Attributes and Style Properties that occur in WebCGM DOM as attributes and method parameters are represented in XCF as XML attributes. The values are encoded in the XML attribute strings of XCF exactly as they would be encoded in the WebCGMString type or a WebCGMString sub-type of the corresponding WebCGM DOM parameter or attribute.

5.6 Coordinates and transforms

5.6.1 Coordinate values — Normalized VDC (NVDC)

In a WebCGM instance, the representation of coordinates (VDC) is influenced by several CGM elements: VDC TYPE, VDC EXTENT, and SCALE MODE. WebCGM requires that SCALE MODE be 'metric', but places few other constraints. Therefore VDC (times some scale factor) are equivalent to millimeters, but otherwise the coordinate system could have a lot of variability: upper-left or lower-left origin, right-handed or left-handed, integer values or real values (floating or fixed), etc.

To simplify working with coordinates, the WebCGM DOM defines and uses a canonical, normalized coordinate system, Normalized VDC (NVDC).

NVDC units are millimeters, in a coordinate system whose origin corresponds to the lower left corner of the VDC extent, with the X axis pointing to the right, and the Y axis pointing up. The following examples illustrate the correspondence between NVDC and VDC values for several WebCGM instances.

Example 1: Simplest possible example, the VDC and the NVDC are identical

- VDC Type: Real
- VDC Extent (lower-left & upper-right corners): (0.,0.) (150.,100.)
- Scale Factor: 'metric', 1.0

The picture's VDC have lower-left origin, X increases to right, Y increases up, picture is 150 mm wide and 100 mm high. The NVDC are identical,: (0.,0.) for lower-left corner, (150.,100.) for upper-right corner. If `(x,y)` are VDC and `(x',y')` are NVDC, then:

- `x' = x`
- `y' = y`

Example 2: The VDC define an upper-left origin, and correspond to a U.S. paper size of 8.5x11.0 inches:

- VDC Type: Real
- VDC Extent (lower-left & upper-right corners): (0.,11.0) (8.5,0.)
- Scale Factor: 'metric', 25.4

In VDC space, the origin is the upper-left corner, X increases to right, Y increases down. In NVDC space, the lower-left corner coordinates (as always) are (0,0) and the upper right corner is (215.9,279.4). If `(x,y)` are VDC and `(x',y')` are NVDC, then:

- `x' = 25.4*x`
- `y' = 279.4 - 25.4*y`

Example 3: In the **general case**, if VDC Extent coordinates are `(xll, yll)`, `(xur, yur)`, and Scale Factor is 'metric', `s`, then `(x',y')` NVDC is derived from `(x,y)` VDC by:

- `x' = sign(xur-xll) * (s) * (x - xll)`
- `y' = sign(yur-yll) * (s) * (y - yll)`

5.6.2 Geometric transform

The following subsections define the conventions and rules associated with the geometric transform, which are implemented in the DOM by methods on the WebCGMAppStructure interface.

5.6.2.1 Transform basic concepts

A geometric transform may be attached to eligible APS nodes by a DOM call or XCF data, and leads to transient visual modification of the displayed image. Eligible APS nodes are: grobject, para, subpara, layer.

Geometric transforms on nodes in the WebCGM object tree, whether the default (Identity) or explicitly specified, compose or combine with the transforms on ancestor and descendant nodes, to define a composite transform — the Current Transformation Matrix — for every node in the tree.

Terminology: WebCGM geometric transforms are defined in the two-dimensional NVDC coordinate space. In principle, any rotate and scale operations can be represented by a 2x2 matrix **M**, which is multiplied by the 2x1 vector represenation for of a point **p** to apply the transform. A translation by **d=(dx,dy)** is performed by adding **d** to **p**.

WebCGM uses the homogeneous coordinate system to define and describe the effect of transforms. In this system, the matrix representation **M** is 3x3, with the six unique numbers associated with rotation, scale, and translation comprising the first two rows, and the third row always being (0,0,1). The point p is represented by the 3x1 vector (x,y,1).

```
a  c  e

b  d  f

0  0  1
```

For convenience, this matrix form **M** will be referred to as: **[a b c d e f]**. The application of the transform **M** to the point **p** is then defined by:

```
p' = M * p
```

The following definitions specify how to form **M** cooresponding to basic operations such as translate, rotate, scale:

1. translate by **(dx,dy)**: **M = [1 0 0 1 dx dy]**
2. rotate around the origin **(0,0)** by angle **a**: **M = [cos(a) sin(a) -sin(a) cos(a) 0 0]**
3. scale around the origin **(0,0)** by factors **sx** and **sy**: **M = [sx 0 0 sy 0 0]**

Successive basic operations are performed by left-multiplying the matrices corresponding to the operations. For example, a translation by Mt followed by a rotation by Ma is performed by:

```
p' = Ma * Mt * p
```

As another example, a rotation of angle **a** about an arbitrary point **c = (cx,cy)** is performed by the sequence of operations:

- **M1**: translate(-cx,-cy)
- **M2**: rotate(a)
- **M3**: translate(cx,cy)

Forming **M1**, **M2**, and **M3** by the above rules, then the matix **M** for the rotation by **a** about **(cx,cy)** is:

```
M = M3 * M2 * M1
```

The WebCGMAppStructure Interface contains methods for applying scale, rotate, translate, and general matrix transforms to Application Structures. Each of the methods has a parameter, *replace*, whose values may be '`combine`' or '`replace`'. These have the following meanings:

- `replace`: replace any existing, explicitly-defined transform on the APS with the new transform.
- `combine`: combine the new transform with any existing explicitly-specified transform by left multiplying. If **Mn** is the newly specified transform, and **Me** is the existing explicitly-specified transform, then the new explicitly-specified transform **M** is:

  ```
  M = Mn * Me
  ```

5.6.2.2 Composition of transforms in the object tree

Assume that an object tree has this structure, where A contains B, which contains C and D:

APS-A

....APS-B

........APS-C

........APS-D

Placing a transform on node A transforms all of the geometry within A, including the contents of B, C, and D. But it doesn't supersede any transform that might be on B, C, and D. Rather, it combines with them — you post-multiply the matrix representations so that the various contents in the tree are transformed as follows:

Ma — inside APS-A, but outside APS-B.

Ma*Mb — inside APS-B, but outside APS-C and APS-D.

Ma*Mb*Mc — inside APS-C.

Ma*Mb*Md — inside APS-D.

The resultant composite transform as defined in these illustrations is known as the ***Current Transformation Matrix*** *(CTM)* of the node. Every node in the object tree conceptually has an associated CTM, even if it is just the Identity matrix.

There is a parameter associated with the WebCGM geometric transform that determines, when the DOM method specifies a 'transform' on a node in the object tree, whether it is defined in '`replace`' mode or '`combine`' mode. But once that transform-specifying method has been executed, then there is some well-defined explicitly-specified transform on that node, and it combines with its ancestors and descendants according to the usual rules to compute the CTM for each node.

5.7 Fundamental Interfaces

The interfaces within this section are considered fundamental, and must be fully implemented by all conforming implementations of the WebCGM DOM. The WebCGM DOM presents WebCGM documents as a hierarchy of **`WebCGMNode`** objects that also implement other, more specialized interfaces. Some of the node types may have child nodes of various types, and others are leaf nodes that cannot have anything below them in the WebCGM document structure. WebCGM has the following node types:

`WebCGMMetafile` — contains a list of WebCGMPicture nodes.

`WebCGMPicture` — may contain child WebCGMAppStructures or child XML metadata nodes.

`WebCGMAppStructure` — may contain child WebCGMAppStructures or child XML metadata nodes.

WebCGMAttr — no children.

The WebCGM DOM also specifies several other interfaces to facilitate access to WebCGM attributes. The **GetWebCGMDocument** interface is the medium between the host environment and the WebCGM functionality.
The **WebCGMNodeList** interface enables the handling of ordered lists of WebCGMNodes. The **WebCGMEvent** interface provides contextual information regarding mouse events. WebCGMNodeList objects in the DOM are live; that is, changes to the underlying document structure are reflected in all relevant NodeList objects. For example, if a DOM user gets a WebCGMNodeList object containing the children of an WebCGMAppStructure, then changes one of its children in the tree, all changes are reflected in the NodeList objects and in fact to all references to that Node in NodeList objects.

The WebCGMPicture node is the root of the document tree for DOM and inheritance model purposes:

- While it may seem natural that WebCGMMetafile should inherit from WebCGMNode, the WebCGM profile takes the view that the metafile is simply a placeholder for generic data formatting information for the picture(s) in a metafile.
- The inheritance root (see inheritance model) should correspond to the webcgm root element of the XML Companion File (XCF), and the XCF design has determined that companion files should be per-picture.
- Although WebCGM is a one-picture profile, extensibility to multi-picture profiles is already designed into the interfaces, and that extensibility relies on the DOM-tree (inheritance) root being the picture.

5.7.1 Common definitions

5.7.1.1 Exception WebCGMException

WebCGM operations only raise exceptions when an operation is impossible to perform.

IDL Definition

```
exception WebCGMException {
  unsigned short   code;
};

// ExceptionCode
const unsigned short      INDEX_SIZE_ERR                = 1;
const unsigned short      WEBCGMSTRING_SIZE_ERR         = 2;
const unsigned short      INVALID_CHARACTER_ERR         = 3;
const unsigned short      NO_DATA_ALLOWED_ERR           = 4;
const unsigned short      NO_MODIFICATION_ALLOWED_ERR   = 5;
const unsigned short      NOT_SUPPORTED_ERR             = 6;
const unsigned short      INVALID_ACCESS_ERR            = 7;
const unsigned short      FILE_NOT_FOUND_ERR            = 8;
const unsigned short      FILE_INVALID_ERR              = 9;
```

Basic example

EXAMPLE:

This simple example shows how to catch a WebCGMException from HTML & ECMAScript. In a WebCGM-DOM enabled browser, correct execution of this HTML-ECMAScript code should cause an alert box to appear that an attempt to change a readonly attribute is made.

```
<html>
<head>
<title>Example, WebCGMException interface</title>
<script type="text/ecmascript">
  function causeException(evt) {
    alert("Try to change the readonly attribute layername");
    try {
      var webcgm = document.getElementById('ivx1').getWebCGMDocument();
```

```
      var pic = webcgm.firstPicture;
      var obj = pic.getAppStructureByID("A");
      obj.setAppStructureAttr("layername","anotherName");
    } catch (e) {
      alert("Catch the exception: " + e.description);
    }
  }
</script>
</head>

<body onload="causeException()">
<table border="1" rules="cols">
  <tr style="text-align:center;">
    <th>Example, causeException interface</th>
  </tr>
  <tr>
    <td><object id="ivx1" data="ex_Exception.cgm"
               type="image/cgm;Version=4;ProfileId=WebCGM"
               width="480" height="360"></object>
    </td>
  </tr>
</table>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

Definition group ExceptionCode

An integer indicating the type of error generated.

Defined Constants

INDEX_SIZE_ERR; if index or size is negative, or greater than the allowed value.

DOMSTRING_SIZE_ERR; if the specified range of text does not fit into a WebCGMString.

INVALID_CHARACTER_ERR; if an invalid or illegal character is specified, such as in an XML name.

NO_DATA_ALLOWED_ERR; if data is specified for a node which does not support data.

NO_MODIFICATION_ALLOWED_ERR; if an attempt is made to modify an object where modifications are not allowed.

NOT_SUPPORTED_ERR; if the implementation does not support the requested type of object or operation.

INVALID_ACCESS_ERR; if a parameter or an operation is not supported by the underlying object.

FILE_NOT_FOUND_ERR; if the reference document could not be accessed

FILE_INVALID_ERR; if the reference document was not well-formed or was in error.

5.7.1.2 Interface WebCGMRect

IDL Definition

```
interface WebCGMRect {
  attribute float xll;  // x coordinate of the lower-left corner
  attribute float yll;  // y coordinate of the lower-left corner
  attribute float xur;  // x coordinate of the upper right corner
  attribute float yur;  // y coordinate of the upper right corner

  WebCGMRect union(in WebCGMRect r);

};
```

Attributes

xll of type float

x coordinate of the lower-left corner of the rectangle.

yll of type float

y coordinate of the lower-left corner of the rectangle.

xur of type float

x coordinate of the upper-right corner of the rectangle.

yur of type float

y coordinate of the upper-right corner of the rectangle.

Methods

union

Computes the union of the current rectangle with the input rectangle r. Returns resulting new WebCGMRect.

Parameters

r of type WebCGMRect

The rectangle used to calculate the union with the current rectangle.

Return value

WebCGMRect; The resulting rectangle.

Exceptions

No exceptions.

5.7.1.3 Interface WebCGMMatrix

IDL Definition

```
interface WebCGMMatrix {
  attribute float a;
  attribute float b;
  attribute float c;
  attribute float d;
  attribute float e;
  attribute float f;

  WebCGMMatrix multiply (in WebCGMMatrix m);
  WebCGMMatrix inverse(); // raises exception
  WebCGMMatrix translate( in float x, in float y );
  WebCGMMatrix scale( in float sx, in float sy );
  WebCGMMatrix rotate( in float angle, in float rx, in float ry );

};
```

Attributes

a, b, c, d, e, f of type float

The a-f components of the matrix as defined in "Transform basic concepts"

Methods

multiply

Performs matrix multiplication. This matrix is post-multiplied by another matrix, returning a new matrix.

```
Example: newMatrix = thisMatrix x m.
```

Parameters

m of type WebCGMMatrix

in WebCGMMatrix m The matrix which is post-multiplied to this matrix.

Return value

WebCGMMatrix; The resulting matrix.

Exceptions

No exceptions.

inverse

Returns the inverse as a new matrix.

Parameters

No parameters.

Return value

WebCGMMatrix; The resulting matrix.

Exceptions

WebCGMException; INVALID_ACCESS_ERR: Raised if this matrix is not invertable.

translate

Post-multiplies a translate transform on the current matrix and returns a new matrix. Result: newMatrix = thisMatrix x translateMatrix.

Parameters

x, y of type float

The distances to translate respectively in the x-direction and y-direction. Units are NVDC.

Return value

WebCGMMatrix; The resulting matrix.

Exceptions

No exceptions.

scale

Post-multiplies a (possibly) non-uniform scaling transform on the current matrix and returns a new matrix. Result: newMatrix = thisMatrix x scaleMatrix.

Parameters

sx, sy of type float

The scale factors to apply respectively in the x-direction and y-direction.

Return value

WebCGMMatrix; The resulting matrix.

Exceptions

No exceptions.

rotate

Post-multiplies a rotation transform on the current matrix and returns a new matrix. The rotation transform allows for a point of rotation to be specified. Result: newMatrix = thisMatrix x translate(rx,ry) x rotate(angle) x translate(-rx,-ry).

Parameters

angle of type float

The rotation angle. Positive values correspond to counter-clockwise in NVDC space.

rx, ry of type float

Respectively, the x-coordinate and the y-coordinate of the rotation point, in NVDC.

Return value
WebCGMMatrix; The resulting matrix.
Exceptions
No exceptions.

5.7.2 Interface GetWebCGMDocument

Since WebCGM documents are often embedded within a host document such as XHTML, WebCGM user agents are required to implement the **GetWebCGMDocument** interface for the element which references the WebCGM document (e.g., the 'object' tag).

IDL Definition

```
interface GetWebCGMDocument {
  WebCGMMetafile              getWebCGMDocument ( ) raises ( WebCGMException );
  WebCGMString                getAppName();
  WebCGMString                getAppVersion();
};
```

Basic example
See any of the examples in the other interface subsections. They all use and illustrate getWebCGMDocument.

Attributes
No defined attributes.

Methods
getWebCGMDocument
Returns the WebCGMMetafile object for the referenced WebCGM document. If no WebCGM document is open in the viewer, the WebCGMMetafile object will still be returned.
Parameters
No parameters.
Return value
WebCGMMetafile; The WebCGMMetafile object for the referenced WebCGM document.
Exceptions
WebCGMException; NOT_SUPPORTED_ERR: No WebCGMMetafile object is available.
getAppName
Returns the name of the viewer application.
Parameters
No parameters.
Return value
WebCGMString; The name of the viewer application.
Exceptions
No exceptions.
getAppVersion
Returns the version of the viewer application.
Parameters
No parameters.
Return value
WebCGMString; The version of the viewer application.
Exceptions
No exceptions.

5.7.3 Interface WebCGMMetafile

The WebCGMMetafile interface is the entry point to the entire WebCGM document. The interface exposes information regarding the metafile and provides access to the first WebCGMPicture of the WebCGM document.

IDL Definition

```
interface WebCGMMetafile {
  readonly attribute WebCGMString  metafileDescription;
  readonly attribute WebCGMPicture firstPicture;
  readonly attribute WebCGMString  metafileID;
  readonly attribute unsigned short metafileVersion;
           attribute WebCGMString  src;
  void               addEventListener(in WebCGMString type,
                                      in WebCGMEventListener listener);
  void               removeEventListener(in WebCGMString type,
                                         in WebCGMEventListener listener));
  void               setRedraw(in WebCGMString value);
};
```

Basic example

EXAMPLE:

This simple example shows how to use a method of the WebCGMMetafile interface from HTML & ECMAScript. In a WebCGM-DOM enabled browser, correct execution of this HTML-ECMAScript code should display the metafileID parameter of the BEGIN METAFILE element in a message below the picture.

```
<html>
<head>
<title>Example, Metafile interface</title>
<script type="text/ecmascript">
  function metafileInfo() {
    try {
      var webcgm = document.getElementById('ivx1').getWebCGMDocument();
      document.getElementById("_1").firstChild.data = "The metafileID is \"" + webcgm.
metafileID + "\"";
    } catch (e) {
      alert(e.description);
    }
  }
</script>
</head>

<body onload="metafileInfo()">
<table border="1" rules="cols">
  <tr style="text-align:center;">
    <th>Example, WebCGMMetafile interface</th>  </tr>
  <tr>
    <td><object id="ivx1" data="ex_Metafile.cgm"
                          type="image/cgm;Version=4;ProfileId=WebCGM"
                          width="480" height="360"></object>
    </td> </tr>
</table>
<p id="_1">Metafile ID is:</p>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

Attributes

metafileDescription of type WebCGMString, readonly

Returns the Metafile Description of the WebCGM document, which is a string consisting of QUOTE-delimited substrings, as defined in the WebCGM PPF. For example: "ProfileId:WebCGM""ProfileEd:2.1""Source:A software vendor""Date:20040602""ColourClass:monochrome". Also as specified by the WebCGM PPF, a valid **metafileDescription** will always contain the ProfileId: and the ProfileEd:, other information such as Source, ColourClass etc... is optional. If no

WebCGM document is open in the viewer, an empty string is returned.

firstPicture of type WebCGMPicture, readonly

Returns the first WebCGMPicture element of the WebCGM document. Subsequent WebCGMPictures can be accessed using the WebCGMPicture interface. A WebCGM document (version 2.0 and later) contains exactly one WebCGMPicture. If no WebCGM document is open in the viewer, null is returned.

metafileID of type WebCGMString, readonly

Returns the Metafile Identifier, which is the parameter of the BEGIN METAFILE element in the CGM document. If no WebCGM document is open in the viewer, an empty string is returned.

metafileVersion of type unsigned short, readonly

Returns the Metafile Version of the WebCGM document. If no WebCGM document is open in the viewer, the value zero is returned

src of type WebCGMString

The IRI of the current document. *On setting*, the new document pointed to by the IRI is loaded by the user agent. The user agent must fully parse the fragment identifier (if any) in the IRI and execute the indicated behavior. The "Picture behaviors" rule does apply to the 'src' attribute — if a IRI fragment contains a picBehavior, the viewer shall ignore the picBehavior. If the CGM resource pointed to by the IRI is currently loaded for the object, the user agent shall not reload the CGM (similar to the specification of a same-CGM IRI for the _replace behavior on a CGM-to-CGM link.) *On retrieval*, if no WebCGM document is open in the viewer, an empty string is returned.

EXAMPLE. The 'src' attribute is an IRI, with possible fragment containing object selection and object behavior terms. It is *not* a full 'linkuri' (APS Attribute) data record, and any fragment does *not* contain picture behavior terms. So to open myCGM.cgm using (ECMAscript) DOM calls that reference an HTML `<object>` element with ID of 'myObjElt':

Correct examples:

```
document.getElementById('myObjElt').getWebCGMDocument().src= 'myCGM.cgm#myId';

[...].getWebCGMDocument().src= 'myCGM.cgm#id(myId,full)';
```

Incorrect:

```
[...].getWebCGMDocument().src= 'myCGM.cgm#"myId" "myTitle" "_blank"'

[...].getWebCGMDocument().src= "myCGM.cgm#id(myId).picseqno(1,_blank)"
```

Methods

addEventListener

This method allows the registration of event listeners on the WebCGMMetafile. If a WebCGMEventListener is added to the WebCGMMetafile while it is processing an event, it will not be triggered by the current actions. If multiple identical WebCGMEventListeners are registered on the WebCGMMetafile with the same parameters the duplicate instances are discarded. They do not cause the WebCGMEventListener to be called twice.

Note: Although all WebCGMEventListeners on the WebCGMMetafile are guaranteed to be triggered by any event which is received, no specification is made as to the order in which the WebCGMMetafile will receive the event with regards to the other WebCGMEventListeners on the WebCGMMetafile.

Parameters

type of type WebCGMString

The event type for which the user is registering, (for example: "click", "mouseover").

listener of type WebCGMEventListener

The listener parameter takes an interface implemented by the user which contains the methods to be called when the event occurs.

Return value

No return value.

Exceptions

No exceptions.

removeEventListener

This method allows the removal of event listeners on the WebCGMMetafile. If an WebCGMEventListener is removed from the WebCGMMetafile while it is processing an event, it will not be triggered by the current actions. WebCGMEventListeners can never be invoked after being removed. Calling removeEventListener with arguments which do not identify any currently registered WebCGMEventListener on the WebCGMMetafile has no effect.

Parameters

type of type WebCGMString

Specifies the event type of the WebCGMEventListener being removed (for example: "click", "mouseover").

listener of type WebCGMEventListener

Indicates the WebCGMEventListener to be removed.

Return value

No return value.

Exceptions

No exceptions.

setRedraw

Disables or enables the redrawing of the metafile. The default value is enableAll. Note when this interface is used and set to enableAll a redraw of the picture automatically will occur. Care should be used when disabling the redrawing function to insure it is set back to enableAll

Parameters

value of type WebCGMString

Specifies the redraw mode of the metafile, {enableAll | disableAll}.

Return value

No return value.

Exceptions

No exceptions.

5.7.4 Interface WebCGMNode

The WebCGMNode interface is the base datatype of the WebCGM Document Object Model. The WebCGMNode object is the basis of several other interfaces. the WebCGMNode object is the basis of several other interfaces, including interfaces to WebCGM specific elements (eg: WebCGMAppStructure), and to non-WebCGM elements such as Metadata nodes. The WebCGMNode interface specifies the attributes and methods to perform simple and generic tree traversal. For these attributes and methods in particular, APS of type 'grnode' are DOM-visible and DOM-accessible, unlike most other DOM interfaces.

IDL Definition

```
interface WebCGMNode {
  const unsigned short PICTURE_NODE          = 1;
  const unsigned short APP_STRUCTURE_NODE    = 2;
  const unsigned short XML_METADATA_NODE     = 3;
  const unsigned short TEXT_NODE             = 4;
  const unsigned short ATTR_NODE             = 5;

  readonly attribute WebCGMString        nodeName;
  readonly attribute WebCGMString        nodeValue;
                                         // raises(WebCGMException) on retrieval

  readonly attribute unsigned short      nodeType;
  readonly attribute WebCGMNode          parentNode;
  readonly attribute WebCGMNodeList      childNodes;
  readonly attribute WebCGMNode          firstChild;
  readonly attribute WebCGMNode          lastChild;
  readonly attribute WebCGMNode          previousSibling;
  readonly attribute WebCGMNode          nextSibling;
  readonly attribute WebCGMPicture       ownerPicture;
  boolean                                hasChildNodes();
  boolean                                hasAttributes();
  readonly attribute WebCGMNodeList      attributes;

  readonly attribute WebCGMString        namespaceIRI;
  readonly attribute WebCGMString        prefix;

  readonly attribute WebCGMString        localName;

  WebCGMString                           getAttributeNS(in WebCGMString namespaceIRI,
                                                        in WebCGMString localName);
  void                                   setAttributeNS(in WebCGMString namespaceIRI,
                                                        in WebCGMString qualifiedName,
                                                        in WebCGMString value);
  WebCGMNodeList                         getElementsByTagNameNS(in WebCGMString namespaceIRI,
                                                                in WebCGMString localName);
};
```

Basic example

EXAMPLE:

This simple example shows how to use an attribute on the WebCGMNode interface from HTML & ECMAScript. In a WebCGM-DOM enabled browser, correct execution of this HTML-ECMAScript code should display a message below the picture, with a 2nd line indicating nodeType "1".

```
<html>
<head>
<title>Example, WebCGMNode interface</title>
<script type="text/ecmascript">
  function getNodeType(evt) {
    var webcgm = document.getElementById('ivx1').getWebCGMDocument();
    if( webcgm ) {
      var pic = webcgm.firstPicture;
      if( pic ) {
        var elem = document.getElementById('result').lastChild;
        var text = elem.nodeValue;
        text = text + pic.nodeType;
        elem.nodeValue = text;
      }
    }
  }
</script>
```

```
</head>

<body onload="getNodeType()">
<table border="1" rules="cols" width="480">
  <tr style="text-align:center;">
    <th>Example, WebCGMNode interface</th>  </tr>
  <tr>
    <td><object id="ivx1" data="ex_Node.cgm"
               type="image/cgm;Version=4;ProfileId=WebCGM"
               width="480" height="360"></object>
    </td>  </tr>
  <tr style="text-align:left;" >
    <td id="result">The <strong>WebCGMNode.nodeType</strong>
                  value of...<br/><em>getWebCGMDocument().firstPicture</em> is: </td>  </
tr>
</table>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

Definition group NodeType

An integer indicating which type of node this is.

Defined Constants

PICTURE_NODE; the node is a WebCGMPicture.

APP_STRUCTURE_NODE; the node is a WebCGMAppStructure.

XML_METADATA_NODE; the node is XML companion information attached to a CGM element.

TEXT_NODE; the node contains character data.

ATTR_NODE; the node is a WebCGMAttr.

Attributes

nodeName of type WebCGMString, readonly

The name of this node, depending on its type; see the table below.

nodeValue of type WebCGMString, readonly

The value of this node, depending on its type; see the table below.

Exceptions on setting

WebCGMException; NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly and if it is not defined to be null.

Exceptions on retrieval

WebCGMException; DOMSTRING_SIZE_ERR: Raised when it would return more characters than fit in a WebCGMString variable on the implementation platform.

nodetype of type unsigned short, readonly

A code representing the type of the underlying object.

The values of nodeName and nodeValue vary according to the node type as follows:

Interface	nodeName	nodeValue
WebCGMAppStructure	WebCGMAppStructure type: "layer" | "grobject" | "para" | "subpara" | "grnode"	empty string
WebCGMAttr	WebCGMAttr.name	empty string
WebCGMPicture	"#picture"	empty string
Character Data	"#text"	content of the text node
XML Metadata	prefix + localName	empty string

parentNode of type WebCGMNode, readonly

The parent (immediate ancestor node of a node) of this node. All nodes, except WebCGMPicture and WebCGMAttr, may have a parent.

childNodes of type WebCGMNodeList, readonly

A WebCGMNodeList that contains all children of this node. If there are no children, this returns null.

firstChild of type WebCGMNode, readonly

The first child of this node. If there is no such node, this returns null.

lastChild of type WebCGMNode, readonly

The last child of this node. If there is no such node, this returns null.

previousSibling of type WebCGMNode, readonly

The node immediately preceding this node. If there is no such node, this returns null.

nextSibling of type WebCGMNode, readonly

The node immediately following this node. If there is no such node, this returns null.

ownerPicture of type WebCGMPicture, readonly

The WebCGMPicture object associated with this node. When the node is a WebCGMPicture node, this returns null

attributes of type WebCGMNodeList, readonly

A WebCGMNodeList containing all attributes (WebCGM and namespaced) of this node or null if the WebCGMNode doesn't have any attributes. Always null when nodeType is APS_STRUCTURE_NODE and nodeName is 'grnode'. The 'apsid' parameter of the Begin APS element is considered to be an attribute of its APS for DOM purposes, and the 'pictid' parameter of the Begin Picture element is considered to be an attribute of its Picture for DOM purposes. This table summarizes the contents of 'attributes' for the various node types:

Node type	attributes
PICTURE_NODE	pictid (always); NS attributes
APP_STRUCTURE_NODE	apsid (always); NS attributes; APS Attributes
XML_METADATA_NODE	NS attributes
ATTR_NODE	none (always null)
TEXT_NODE	none (always null)

namespaceIRI of type WebCGMString, readonly

The namespace IRI of this node. For example, on the element `foo:someElement`, returns the IRI of the (xmlns) namespace declaration that associates the prefix `foo` with the namespace. This is not a computed value that is the result of a namespace lookup based on an examination of the namespace declarations in scope. It is the namespace IRI given at creation time. This returns empty string if the WebCGMNode is not of type XML_METADATA_NODE or ATTR_NODE.

prefix of type WebCGMString, readonly

The namespace prefix of this node (e.g., foo:elementName, returns "foo"). This returns empty string if the WebCGMNode is not of type XML_METADATA_NODE or ATTR_NODE.

localName of type WebCGMString, readonly

Returns the local part of the qualified name of this node (e.g., foo:elementName, returns "elementName"). This returns empty string if the WebCGMNode is not of type XML_METADATA_NODE or ATTR_NODE.

Methods

hasChildNodes

Returns whether this node has any children.

Parameters

No parameters.

Return value

boolean; true if this node has any children, false otherwise.

Exceptions

No exceptions.

hasAttributes

Returns whether this node has any attributes. (For more information, see the **attributes** attribute.)

Parameters

No parameters.

Return value

boolean; true if this node has any attributes, false otherwise.

Exceptions

No exceptions.

getAttributeNS

Returns the node attribute value by local name and namespace IRI.

Parameters

namespaceIRI of type WebCGMString

The namespace IRI of the attribute to retrieve.

localName of type WebCGMString

The local name of the attribute to retrieve.

Return value

WebCGMString; The WebCGMAttr value as a string, or the empty string if that attribute does not have a specified value.

Exceptions

No exceptions.

setAttributeNS

Adds a new attribute. If an attribute with that name is already present on the node, its value is changed to be that of the value parameter.

Parameters

namespaceIRI of type WebCGMString

The namespace IRI of the attribute to create or alter.

qualifiedName of type WebCGMString

The qualified name of the attribute to create or alter.

value of type WebCGMString

The value to set, in string form.

Return value

No return value.

Exceptions

WebCGMException; INVALID_CHARACTER_ERR: Raised if the specified qualified name contains an illegal character. The legal-character constraints of the qualified name match those of the **attribute name** construct of *XML 1.0 Third Edition*.

getElementsByTagNameNS

Returns a WebCGMNodeList of all the descendant XML companion file elements (application specific metadata) with a given local name and namespace IRI in the order in which they are encountered in a preorder traversal of the WebCGMNode tree. Returns null if there are no such elements.

Parameters

namespaceIRI of type WebCGMString

The namespace IRI of the XML elements to match on.

localName of type WebCGMString

The local name of the XML elements to match on.

Return value

WebCGMNodeList; A list of matching XML element nodes.

Exceptions

No exceptions.

5.7.5 Interface WebCGMPicture

The WebCGMPicture interface allows for access to the Application Structures of the WebCGM document. It also specifies how to load and apply an XML Companion File (XCF) to a WebCGM document.

IDL Definition

```
interface WebCGMPicture : WebCGMNode {
  readonly attribute float           width;
  readonly attribute float           height;
  readonly attribute WebCGMString    pictid;
  boolean            applyCompanionFile(in WebCGMString fileIRI);
  WebCGMAppStructure getAppStructureById(in WebCGMString apsId);
  WebCGMNodeList     getAppStructuresByName(in WebCGMString apsName);
  void               highlight(in WebCGMNodeList nodes,
                               in WebCGMString type);
  void               clearHighlight();
  void               setPictureVisibility(in WebCGMString visibility);
  void               setStyleProperty(in WebCGMString style,
                                      in WebCGMString value);
  void               reloadPicture();
  void               setView (in WebCGMRect viewRect);
  WebCGMString       getStyleProperty(in WebCGMString style);
};
```

Basic example

EXAMPLE:

This simple example shows how to use a method of the WebCGMPicture interface from HTML & ECMAScript. In a WebCGM-DOM enabled browser, execution of this HTML-ECMAScript code should cause the initial view of the technical illustration to appear with a blue-gray background, instead of a white background:

```
<html>
<head>
<title>Example, WebCGMPicture interface</title>
<script type="text/ecmascript">
  function changeBackground(evt) {
    var webcgm = document.getElementById('ivx1').getWebCGMDocument();
    if( webcgm ) {
      var pic = webcgm.firstPicture;
      if( pic ) {
        pic.setStyleProperty("background-color","#A0A0D0");
      }
    }
  }
</script>
</head>

<body onload="changeBackground()">
<table border="1" rules="cols">
  <tr style="text-align:center;">
    <th>Example, WebCGMPicture interface</th>
  </tr>
  <tr>
    <td><object id="ivx1" data="ex_Picture.cgm"
               type="image/cgm;Version=4;ProfileId=WebCGM"
               width="480" height="360"></object>
    </td>
  </tr>
</table>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

Attributes

width of type float, readonly

Represents the WebCGMPicture width in millimeters. Please refer to Coordinate Values section for more information.

height of type float, readonly

Represents the WebCGMPicture height in millimeters. Please refer to Coordinate Values section for more information.

pictid of type WebCGMString, readonly

Represents the WebCGMPicture id, which is the id parameter in the BEGIN PICTURE element in the CGM document.

Methods

applyCompanionFile

Reads an XML Companion File (XCF) into the user agent's object model. If application-specific metadata is found in the companion file (in the form of namespace attributes and namespace children elements), the user agent will create new namespace application structures as children of existing WebCGM Application Structures within it's object model.
This information will then be accessible using methods found on this WebCGMPicture interface, on WebCGMAppStructure and on WebCGMNode. If the `fileIRI` parameter of this method is a relative IRI, then that relative IRI is resolved similarly to resolving relative IRIs for XCF resources referenced in the WebCGM IRI fragment syntax, i.e., the IRI is resolved relative to location of the CGM resource to which the XCF resource is a companion.

Please refer to the *Relationship with XML companion file* section for more discussion.

Parameters

fileIRI of type WebCGMString

The file name and location of the XML companion file to load and apply into the object model.

Return value

boolean; true if the implementation was able to load and parse the XML companion file into memory as requested; false otherwise.

Exceptions

WebCGMException; FILE_NOT_FOUND_ERR; if the referenced document could not be accessed.

WebCGMException; FILE_INVALID_ERR: if the referenced document was not well-formed or in error.

getAppStructureById

Returns the Application Structure whose ID is given by `apsId`. If no such Application Structure exists, returns null. Returns null if `apsid` corresponds to an APS of type 'grnode'. Behavior is not defined if more than one element has this ID. Only WebCGM Application Structures may be retrieved using getAppStructureById, it does not retrieve foreign namespace elements (application-specific metadata elements).

Parameters

apsId of type WebCGMString

The unique id value for an Application Structure.

Return value

WebCGMAppStructure; a WebCGMAppStructure object containing the Application Structure with the matching id.

Exceptions

No exceptions.

getAppStructuresByName

Returns the list of Application Structures whose names are given by apsName in the order in which they are encountered in a depth-first-order traversal of the WebCGMPicture tree. If no such Application Structures exists, returns null. Only WebCGM Application Structures may be retrieved using getAppStructuresByName, it does not retrieve foreign namespace elements (application-specific metadata elements).

Parameters

apsName of type WebCGMString

A non-unique name value for an Application Structure.

Return value

WebCGMNodeList; A WebCGMNodeList object containing all the matching Application Structure WebCGMNodes.

Exceptions

No exceptions.

highlight

Highlights a collection of Application Structures (APSs). WebCGM also allows for highlighting of APSs using the IRI fragment syntax. The exact method of highlighting is viewer dependent. The highlight method provides a way for WebCGM script writers to highlight APSs in the same way a IRI fragment would. It also allows for highlighting of entire layers. Highlighting is not defined for WebCGMPicture nodes or XML Metadata nodes. APS of type 'grnode' are not valid in the node list, and shall cause no change to the viewed image or the DOM tree.

Parameters

nodes of type WebCGMNodeList

A WebCGMNodeList of APP_STRUCTURE_NODEs to highlight.

type of type WebCGMString

Denotes a behavior identical to the corresponding highlighting object behavior keywords of the fragment syntax. Values: { add | new }.

Return value

No return value.

Exceptions

No exceptions.

clearHighlight

Clears highlighting for all currently highlighted APSs in the picture. The behavior is identical to the **special-form object behavior** fragment, `id(*,clearHighlight)`, that is defined in the enumeration of behaviors of the fragment syntax.

Parameters

None.

Return value

No return value.

Exceptions

No exceptions.

setPictureVisibility

Sets the visibility on or off for the whole picture. Note that for the purposes of the inheritance model, the Picture node behaves like an Application Structure, and the visibility as set by this method behaves like the 'visibility' APS Attribute.

Parameters

visibility of type WebCGMString

Value for the visibility of the picture, {on | off}.

Return value

No return value.

Exceptions

No exceptions.

setStyleProperty

Set a style property at the picture level by name.

The following table and text describe in more detail each of the style properties, their scopes and allowed values:

Rules & encodings for DOM manipulation of style properties

Style Property Name	Picture level	APS level	Attribute value(s)	Initial value	Example
background-color	yes	no	absolute RGB or relative intensity (0..100%)	100%	"#000000" or "75%"
text-size	yes	yes	absolute NVDC or relative scale (both > 0)	100%	"225%"
fill-color	yes	yes	absolute RGB or relative intensity (0..100%)	100%	"#FF0000" or "75%"
intensity	yes	yes	intensity (0..100%)	100%	"75%"
stroke-color	yes	yes	absolute RGB or relative intensity (0..100%)	100%	"#FF0000" or "75%"
stroke-weight	yes	yes	absolute NVDC or relative scale (both > 0)	100%	"225%"
text-color	yes	yes	absolute RGB or relative intensity (0..100%)	100%	"#FF0000" or "75%"
text-font	yes	yes	WebCGMString	"metafile"	"Helvetica"
raster-intensity	yes	yes	relative intensity (0..100%)	100%	"75%"
stroke-type	yes	yes	name of line type or a negative integer	1	"solid" or "-3"
stroke-offset	yes	yes	relative distance (0..100%)	0%	"25%"
interior-style	yes	yes	integer value (0,1,2,3,4,6)	0	"4"
hatch-index	yes	yes	index value (1..6 and negative)	1	"1" or "-2"
pattern-index	yes	yes	index value (>0)	1	"2"
edge-visibility	yes	yes	"on" or "off"	"on"	"off"
fill-offset	yes	yes	NVDC point	"0. 0."	"0.2 0.5"

Common specifications. The following common specifications, related to the inheritance model, apply to all of the Style Properties:

- **Inherited: yes**: Each Style Property may be inherited.
- **Value "inherit"**: Each Style Property may take the value "inherit", in addition to the values listed for the individual property.
- **Computed Value**: The Computed Value of each Style Property is the same as the specified value, with the exception of the elimination of the value "inherit" as specified in the inheritance model.

Units in the table. RGB colors are expressed as hexadecimal values. Relative scale values are expressed as a positive or non-negative number (depending on the property) followed by a '%' unit designator. Relative values of some properties can exceed 100%. Relative intensity values are expressed as a number followed by a '%' unit, ex: "75%". Relative intensity values cannot exceed 100%.

Color representation. Absolute RGB colors are expressed using a hexadecimal representation for all three RGB channels, #RRGGBB. Examples of colors expressed in hexadecimal representation: red is expressed as #FF0000, and cyan which uses both full green and full blue is expressed as #00FFFF. The representation must be exactly 6 digits, 2 each for R, G, and B. Shorthand hexadecimal notation, e.g., the 3-digit #RGB notation, is not supported in this specification.

Replacement mode. When Style Properties have values of "%" (percent), the respective attribute value used for display is adjusted by applying the appropriate formulae to the attribute values in the metafile (for the appropriate target object).
For example, stroke-weight of 60% means that the metafile-defined LINE WIDTH and EDGE WIDTH attributes are multiplied by 0.6. Successive setting of the same Style Property replaces any previous setting of the same Style Property (rather than accumulating with it). So, for example, stroke-weight 60% followed later by stroke-weight 40% results in stroke-weight 40%, not stroke-weight 24%.

Order counts. Some Style Properties have overlapping effects. For example, intensity and fill-color both affect the color of filled areas. When both properties are defined for a target APS, the latter definition supersedes and replaces the earlier definition. So for example intensity 40% followed by fill-color 60% results in 60% fill color, while fill-color #FF0000 followed by intensity 40% results in fill-color 40% (of the filled-areas colors in the metafile, for the target object).

Style Property Definitions. The following are the detailed functional definitions of each of the Style Properties:

background-color is the color of the rendering surface for the entire picture, on which all elements are drawn. It corresponds to the BACKGROUND COLOUR attribute of the CGM standard. Example: a value of #000000 for the background-color style property will override what is in the WebCGM instance, and display a black background for all elements to render over.

text-size redefines the size of all text in the target object. If text-size is "%", then it adjusts the text restriction boxes (heights and widths) and the CGM CHARACTER HEIGHT attribute by that amount. If the text-size value is NVDC, for each text element in the target object, compute the ratio (effectively, a percent) of the new NVDC value and the restriction box height, and apply the resulting ratio as would be done for the same "%" value.

fill-color is the style property applied to a closed area inside the path of a shape. It corresponds to the CGM attribute FILL COLOUR, and will override the current values of that attribute within the target object (APS or picture), if the fill-color Style Property is applied to object.

intensity is a way to make the current color fade towards white. An intensity value of 0% applied to an Application Structure (APS) will make its contents completely white while a value of 100% will keep the current colors intact. The intensity equation is as follows:

```
normalizedNewRed = 1 - intensity * (1 - normalizedOldRed)
normalizedNewGreen = 1 - intensity * (1 - normalizedOldGreen)
normalizedNewBlue = 1 - intensity * (1 - normalizedOldBlue)
```

Example: Here is an example of the computations when applying an intensity of 40% to the color orange #FFA500:

```
normalizedNewRed = 1 - 0.4 * (1 - 1) = 1
normalizedNewGreen = 1 - 0.4 * (1 - 0.647) = 0.859
normalizedNewBlue = 1 - 0.4 * (1 - 0) = 0.5
```

The new color is %FFDB99.

Setting a relative intensity value is allowed on a number of individual style properties, see table above. The 'intensity' style property, however, represents a convenience property that simultaneously controls the intensity value of the following four properties: fill-color, stroke-color, text-color and raster-intensity.

stroke-color defines the color for the lines and edges within the target object (APS or picture) to which the property is applied. Stroke-color overrides the CGM attributes LINE COLOUR and EDGE COLOUR. This style property will apply an absolute or a relative intensity color change to metafile-defined values of those CGM attributes within the target object.

stroke-weight redefines the thickness of the pen strokes for drawing of lines and edges within the target object (APS or picture) to which it is applied. Stroke-weight overrides CGM attributes LINE WIDTH and EDGE WIDTH. This stroke-weight property can apply a relative scale change to the metafile-defined value of those attributes, or can provide an absolute (NVDC) replacement for those current values.

text-color redefines the color for the graphical text within the target object (APS or picture) to which the property is applied. Text-color overrides the CGM attribute TEXT COLOUR. This style property will apply an absolute or a relative intensity color change to metafile-defined value of that CGM attribute within the APS.

text-font specifies a replacement font for all text in the target object. If the characters that are needed for all text in the target object are available in the specified replacement font, and if the specified font is available, then use it for all text in the target object. Otherwise, ignore the specified replacement font. The initial value of text-font, which is the reserved keyword "metafile", means that the font specifications of the metafile are used.

raster-intensity is a way to make the current color fade towards white in a raster element. It applies to the colors within CELL ARRAY, TILE, and BITONAL TILE elements within the target object (APS or picture) to which it is applied. An intensity value of 0% applied to an Application Structure (APS) will make its raster contents completely white while a value of 100% will keep the current raster colors intact. The equations for computing new color values are the same as for the **intensity** property, above.

stroke-type defines the line type within the target object (APS or picture) to which the property is applied. stroke-type overrides the CGM attribute elements LINE TYPE and EDGE TYPE. Valid values are: integers 1-6 (which correspond to solid, dash, dot, dash-dot, dash-dot-dot), integers 6-15 (the registered or user defined values that are defined within the WebCGM.

stroke-offset defines the percentage of the first cycle of the stroke type that is omitted when starting to draw a non-solid stroke. Stroke offset overrides the CGM attribute elements LINE TYPE INITIAL OFFSET and EDGE TYPE INITIAL OFFSET within the target object (APS or picture).

interior-style determines which style of interior is used to draw a filled-areas. It corresponds to the CGM attribute element INTERIOR STYLE and will override the current values of that attribute within the target object (APS or picture). The valid Style Property values {0,1,2,3,4,6} corresponding respectively to {‘hollow', ‘solid', ‘pattern', ‘hatch', ‘empty', ‘‘interpolated'}.

hatch-index determines which hatch to use within filled-area elements of the target object (APS or picture). Hatch-index corresponds to the CGM attribute element HATCH INDEX and overrides the current values of that attribute within the target object (APS or picture). It must refer to a CGM:1999 pre-defined hatch index or a negative hatch index that has been defined with a HATCH STYLE DEFINITION with the WebCGM. Note: valid pre-defined hatch indexes are 1-6 (corresponding to: horizontal, vertical, positive slope, negative slope, horizontal/vertical cross, positive/negative slope cross)

pattern-index determines which of pattern to use within filled-area elements of the target object (APS or picture). Pattern-index corresponds to the CGM attribute element PATTERN INDEX and overrides the current values of that attribute within the target object (APS or picture). It must refer to a pattern that has been defined with a PATTERN TABLE with the WebCGM.

edge-visibility determines if the edge of filled-area elements of the target object (APS or picture) are drawn. Edge visibility corresponds to the CGM attribute element EDGE VISIBILITY and overrides the current values of that attribute within the target object (APS or picture)

fill-offset sets a reference point for patterns or hatch fills within the target object (APS or picture). Fill-offset corresponds to the CGM attribute element FILL REFERENCE POINT and overrides the current values of that attribute within the target object (APS or picture)

Parameters

style of type WebCGMString

The name of the style property to modify.

value of type WebCGMString

The new value for the given style. Note that "inherit" is a valid value for every Style Property, and per the inheritance model, it has the effect of restoring the style property to its initial (load time) value (as determined by the inheritance model).

Return value
No return value.
Exceptions
No exceptions.

reloadPicture

Notifies the user agent to immediately redraw the entire WebCGMPicture. The user agent will reload the WebCGMPicture while preserving the current user agent's zoom and pan level. The reloading of the WebCGMPicture also discards any existing companion information that may have been loaded into memory via the **applyCompanionFile** method.

Parameters
No parameters.
Return value
No return value.
Exceptions
No exceptions.

getStyleProperty

Retrieves a style property by name on the given Picture. Please refer to the Style Properties Table for more detailed information on retrievable and modifiable Style Properties.

Parameters
style of type WebCGMString

The name of the style property to retrieve.

Return value
WebCGMString; the Style Property value as a string, or the empty string if that attribute does not have an explicitly set value (see the inheritance model for further related discusion). The value may be a Delimited String.
Exceptions
No exceptions.

setView

Sets a view to the specified rectangle expressed in NVDC units.

WebCGMAppStructure::getObjectExtent and WebCGMRect::union can be used to set the view around more than one APS.

Parameters
viewRect of type WebCGMRect

The view rectangle in NVDC.

Return value
No return value.
Exceptions
No exceptions.

5.7.6 Interface WebCGMAppStructure

The WebCGMAppStructure interface offers methods for setting and retrieving Application Structure (APS) attributes. The main methods for accessing Application Structure attributes are getAppStructureAttr and setAppStructureAttr. It is important to note that some attributes, like 'name' and 'linkuri', may have multiple values. In that case, a Delimited String is returned. Delimited String is also used for 'region', which may contain several sub-regions. The WebCGMAppStructure interface has limited impact on APS of type 'grnode'. See the particular methods and attributes for details.

The following table identifies which APS attribute values can be expressed as a Delimited String. Each entry in the table points to the detailed description of the attribute, as it appears in WebCGM content.

Rules & encodings for DOM access to APS Attributes

APS Attribute Name	read/write	Delimited strings	Example
content	yes	no, single string	"car engine transmission"
interactivity	yes	no, single string	"on"
layerdesc	yes	no, single string	"This layer contains English instructions"
layername	readonly	no, single string	"English instructions"

linkuri	yes	yes, multiple 3-tuples possible	"'http://w3.org" "W3C" "_blank"'
name	readonly	yes, multiple names possible	"'firstName" "anotherName"'
region	yes	yes, multiple subregions possible	"'1 0 0 100 100" "1 25 25 75 75"'
screentip	yes	no, single string	"This is a screentip"
viewcontext	yes	no, single string (two corner points)	"0 0 100 100"
visibility	yes	no, single string	"on"

The WebCGMAppStructure interface, like the WebCGMPicture interface, also provides methods for modifying Style Properties at the Application Structure level. For more information about available Style Properties, refer to the Style Properties Table.

IDL Definition

```
interface WebCGMAppStructure : WebCGMNode {
  readonly attribute WebCGMString  apsId;
  readonly attribute unsigned long nameCount;
  readonly attribute unsigned long linkuriCount;

  WebCGMString getAppStructureAttr(in WebCGMString name);
  void         setAppStructureAttr(in WebCGMString name, in WebCGMString value)
                        raises( WebCGMException );
  void         removeAppStructureAttr(in WebCGMString name)
                        raises( WebCGMException );
  void         setStyleProperty(in WebCGMString style, in WebCGMString value);
  WebCGMNodeList toNodeList();
  WebCGMRect   getObjectExtent();
  WebCGMString   getStyleProperty(in WebCGMString style);
  void         translate(in WebCGMString dx, in WebCGMString dy, in WebCGMString replace);
  void         rotate(in WebCGMString angle, in WebCGMString rx, in WebCGMString ry,
in WebCGMString replace);
  void         scale(in WebCGMString sx, in WebCGMString sy, in WebCGMString replace);
  void         setTransform(in WebCGMmatrix matrix, in WebCGMString replace);
  WebCGMMatrix getTransform(in WebCGMString type);
};
```

Examples

EXAMPLE:

This simple example shows how to use methods of the WebCGMAppStructure interface from HTML & ECMAScript. In a WebCGM-DOM enabled browser, correct execution of this HTML-ECMAScript code should display "Layer name is fleet" under the picture.

```
<html>
<head>
<title>Example, WebCGMAppStructure interface</title>
<script type="text/ecmascript">
  function OnBtnDOM() {
    try {
      // Get layernname
      var cgmDoc = document.getElementById("ivx1").getWebCGMDocument();
      var cgmPic = cgmDoc.firstPicture;
      var result = document.getElementById("_1");
      var gr = cgmPic.getAppStructureById("fleet");
      var i = gr.getAppStructureAttr("layername");
      result.firstChild.data = "Layer name is " + i ;
    }
    catch (e) {
      alert("Catch the exception: " + e.description);
```

```
    }
  }
</script>
</head>

<body onload="OnBtnDOM()">
<table border="1" rules="cols">
  <tr style="text-align:center;">
    <th>Example, WebCGMAppStructure interface</th> </tr>
  <tr>
    <td><object id="ivx1" data="ex_AppStructure.cgm"
               type="image/cgm;Version=4;ProfileId=WebCGM"
               width="480" height="360"></object> </td> </tr>
</table>
<p id="_1">Layer name is ...</p>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

EXAMPLE:

The more advanced example in Appendix F shows how to use methods of the WebCGMAppStructure interface from HTML & ECMAScript to perform a regular expression searching based on the contents of APS attributes.

Attributes

apsId of type WebCGMString, readonly

The unique identifier of the Application Structure. Always the empty string if the APS is of type 'grnode'.

nameCount of type unsigned long, readonly

Represents the number of 'name' attribute values present on this Application Structure. Always zero if the APS is of type 'grnode'.

linkuriCount of type unsigned long, readonly

Represents the number of 'linkuri' attribute values present on this Application Structure. Always zero if the APS is of type 'grnode'.

Methods

getAppStructureAttr

Retrieves an Application Structure attribute value by name. Please refer to the Application Structure Attributes table for more detailed information on retrievable and modifiable Application Structure attributes.

Parameters

name of type WebCGMString

The name of the Application Structure attribute to retrieve.

Return value

WebCGMString; the Application Structure attribute value as a string, or the empty string if that attribute does not have an explicitly set value (see the inheritance model for further related discussion). The value may be a Delimited String. Always the empty string if the APS is of type 'grnode'.

Exceptions

No exceptions.

setAppStructureAttr

Adds a new Application Structure attribute. If an attribute with that name is already present in the APS, its value is changed to be that of the value parameter. Please refer to the Application Structure Attributes table for more detailed information on retrievable

and modifiable Application Structure attributes. If the APS is of type 'grnode', this method shall have no effect, neither on the viewed image nor the DOM tree.

Parameters

name of type WebCGMString

The name of the Application Structure attribute to create or alter.

value of type WebCGMString

Value to set in string form. The value may be a delimited string.

Return value

No return value.

Exceptions

WebCGMException; NO_MODIFICATION_ALLOWED_ERR: Raised if the Application Structure Attribute is readonly.

removeAppStructureAttr

Removes an Application Structure attribute. Please refer to the Application Structure Attributes table for more detailed information on retrievable and modifiable Application Structure attributes. If the APS is of type 'grnode', this method shall have no effect, neither on the viewed image nor the DOM tree.

Parameters

name of type WebCGMString

The name of the Application Structure attribute to remove.

Return value

No return value.

Exceptions

WebCGMException; NO_MODIFICATION_ALLOWED_ERR: Raised if the Application Structure Attribute is readonly.

setStyleProperty

Set a style property by name on the given Application Structure. Please refer to the Style Properties Table for more detailed information on style properties. If the APS is of type 'grnode', this method shall have no effect, neither on the viewed image nor the DOM tree.

Parameters

style of type WebCGMString

The name of the style attribute to modify.

value of type WebCGMString

The new value for the given style.

Return value

No return value.

Exceptions

No exceptions.

toNodeList

Creates a new WebCGMNodeList object and inserts the current Application Structure node into the list. The list count is 1.

Parameters

No parameters.

Return value

WebCGMNodeList; A WebCGMNodeList object containing the Application Structure. Always null if the APS is of type 'grnode'.

Exceptions

No exceptions.

getObjectExtent

Retrieves the axis-aligned bounding box rectangle of the Graphical Primitive elements within an APS. The bounding box calculation is based on the abstract locus of the primitives within the APS. Other than text attributes and Style Properties, the calculation is not affected by CGM Primitive Attribute (such as line width) or Control elements, nor by APS Attributes or Style Properties. It is affected by geometric transform — the defining coordinates of the WebCGMRect return value are expressed in NVDC, computed after the application of the Current Transformation Matrix, to the object's contents.

The contribution of text elements to the object extent is conceptually calculated from the containing parallellogram of the displayed text, defined as follows. The length of the side in the text-up direction is the bottomline-to-topline distance of the font, after computation of the effective font size that reflects all text attributes, the height of the Restricted Text box, the Restricted Text Type, and the Style Properties text-size and text-font. The length of the side in the text-baseline direction is the length of the restricted text box if the entire Restricted Text element is contained within the object, or the sum of the glyph widths if only an Append Text element is within the object.

Parameters

No parameters.

Return value

WebCGMRect; the bounding rectangle of the APS, or null if the APS has no Graphical Primitive elements. Always the null if the APS is of type 'grnode'.

Exceptions

No exceptions.

getStyleProperty

Retrieves a style property by name on the given Application Structure. Please refer to the Style Properties Table for more detailed information on retrievable and modifiable Style Properties.

Parameters

style of type WebCGMString

The name of the style property to retrieve. Always the empty string if the APS is of type 'grnode'.

Return value

WebCGMString; the Style Property value as a string, or the empty string if that attribute does not have an explicitly set value (see the inheritance model for further related discusion). The value may be a Delimited String. Always the empty string if the APS is of type 'grnode'.

Exceptions

No exceptions.

translate

Defines on the APS a new geometric transform that consists of a translate operation. Please refer to "Geometric transform" for more detailed information about geometric transforms and the parameters of this method. If the APS is of type 'grnode', this method shall have no effect, neither on the viewed image nor the DOM tree.

Parameters

dx of type WebCGMString

The displacement of the translate operation in the **x** direction, as a number in NVDC.

dy of type WebCGMString

The displacement of the translate operation in the **y** direction (NVDC), as a number in NVDC.

replace of type WebCGMString

How to apply the new transform, {combine | replace}.

Return value

No return value.

Exceptions

No exceptions.

rotate

Defines on the APS a new geometric transform that consists of a rotate operation. Please refer to "Geometric transform" for more detailed information about geometric transforms and the parameters of this method. If the APS is of type 'grnode', this method shall have no effect, neither on the viewed image nor the DOM tree.

Parameters

angle of type WebCGMString

The angle of the rotation operation, in degrees, as a number sub-type. The positive angular direction is counterclockwise in NVDC space.

rx of type WebCGMString

The **x** coordinate (NVDC) of the center point about which the rotation is defined, as a number in NVDC.

ry of type WebCGMString

The **y** coordinate (NVDC) of the center point about which the rotation is defined, as a number in NVDC.

replace of type WebCGMString

How to apply the new transform, {combine | replace}.

Return value

No return value.

Exceptions

No exceptions.

scale

Defines on the APS a new geometric transform that consists of a scale operation. Please refer to "Geometric transform" for more detailed information about geometric transforms and the parameters of this method. If the APS is of type 'grnode', this method shall have no effect, neither on the viewed image nor the DOM tree.

Parameters

sx of type WebCGMString

The **x** coordinate (NVDC) of the reference point about which the scaling operation is defined, as a number in NVDC.

sy of type WebCGMString

The **y** coordinate (NVDC) of the reference point about which the scaling operation is defined, as a number in NVDC.

replace of type WebCGMString

How to apply the new transform, {combine | replace}.

Return value

No return value.

Exceptions

No exceptions.

setTransform

Defines on the APS a new local geometric transform by specifying the contents of the transform matrix. Please refer to "Geometric transform" for more detailed information about geometric transforms and the parameters of this method. If the APS is of type 'grnode', this method shall have no effect, neither on the viewed image nor the DOM tree.

Parameters

matrix of type WebCGMMatrix

The matrix used to determine the new local transform on the node, as described in "Transform basic concepts".

replace of type WebCGMString

How to apply the matrix to determine the new local transform, {combine | replace}, as described in "Transform basic concepts".

Return value

No return value.

Exceptions

No exceptions.

getTransform

Returns current geometric transform information for the APS. Please refer to "Geometric transform" for more detailed information about geometric transforms and the parameters of this method.

Parameters

type of type WebCGMString

Determines which matrix to return, {local | ctm}.

When type is "ctm", the Current Transformation Matrix for this APS is returned. i.e., the accumulation of all transformations that have been defined on this node and all its ancestors, up to the Picture's normalized coordinate system.

When type is "local", the transformation defined on this APS only is returned; ancestor transformations are ignored. Note: the returned matrix must respect previous transform operation modes (replace | combine), if any.

Return value

WebCGMMatrix; The matrix corresponding to the requested tranform on the node.

Exceptions

No exceptions.

5.7.7 Interface WebCGMNodeList

The WebCGMNodeList interface provides the abstraction of an ordered collection of nodes. WebCGMNodeList objects in the WebCGM DOM are live. The index with the WebCGMNodeList starts at 0.

IDL Definition

```
interface WebCGMNodeList {
  readonly   attribute unsigned long count;
  WebCGMNode           item(in unsigned long index);
  WebCGMNode           removeItem ( in unsigned long index )
                         raises( WebCGMException );
  WebCGMNode           appendItem ( in WebCGMNode newItem )
                         raises( WebCGMException );
};
```

Basic example

EXAMPLE:

This simple example shows how to use a method of the WebCGMNodeList interface from HTML & ECMAScript. In a WebCGM-DOM enabled browser, execution of this HTML-ECMAScript code should count the number of planes in the fleet application structure..

```
<html>
<head>
<title>Example, NodeList interface</title>
<script type="text/ecmascript">
  function getNodeList() {
    try {
      var webcgm = document.getElementById('ivx1').getWebCGMDocument();
      var pic = webcgm.firstPicture;
      var mylist = pic.getAppStructureByID("fleet").childNodes;
      document.getElementById("_1").firstChild.data = "The fleet contains "+mylist.count+"
planes." ;
    } catch (e) {
      alert(e.description);
    }
  }
</script>
</head>

<body onload="getNodeList()">
<table border="1" rules="cols">
  <tr style="text-align:center;">
    <th>Example, WebCGMNodeList interface</th>  </tr>
  <tr>
    <td><object id="ivx1" data="ex_NodeList.cgm"
                type="image/cgm;Version=4;ProfileId=WebCGM"
                width="480" height="360"></object>  </td>  </tr>
</table>
<p id="_1">The fleet contains xx planes.</p>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

Attributes

count of type unsigned long, readonly

The number of nodes in the list. The range of valid child node indices is 0 to `count-1` inclusive.

Methods

item

Returns the indexed item in the collection.

Parameters

index of type unsigned long

Index into the collection.

Return value

WebCGMNode; The node of the indexed position in the WebCGMNodeList that is not a valid index., or null if that is not a valid index.

Exceptions

No exceptions.

removeItem

Removes an existing item from the list.

Parameters

index of type unsigned long

The index of the item which is to be removed. The first item is number 0.

Return value

WebCGMNode; The removed item.

Exceptions

WebCGMException; NO_MODIFICATION_ALLOWED_ERR: Raised when the list cannot be modified.

appendItem

Inserts a new item at the end of the list. If newItem is already in a list, it is removed from its previous list before it is inserted into this list.

Parameters

newItem of type WebCGMNode

The item which is to be inserted into the list.

Return value

WebCGMNode; The inserted item.

Exceptions

WebCGMException; NO_MODIFICATION_ALLOWED_ERR: Raised when the list cannot be modified.

5.7.8 Interface WebCGMAttr

The `WebCGMAttr` interface represents an attribute in a XML_METADATA_NODE, a PICTURE_NODE or a APP_STRUCTURE_NODE.

Note that WebCGMAttr objects inherit the WebCGMNode interface, but since they are not actually child nodes of the element they describe, the WebCGM DOM does not consider them part of the document tree. Thus, the WebCGMNode attributes parentNode, previousSibling, and nextSibling have a null value for WebCGMAttr objects.

IDL Definition

```
interface WebCGMAttr: WebCGMNode {
  readonly attribute WebCGMString        name;
           attribute WebCGMString        value;
  readonly attribute WebCGMNode          ownerNode;
};
```

Attributes

name of type WebCGMString, readonly

Returns the name of this attribute. If WebCGMNode.localName is different from the empty string, this attribute is a qualified name.

value of type WebCGMString, readonly

On retrieval, the value of the attribute is returned as a string, see WebCGMNode.getAppStructureAttr(). On setting, is it equivalent to WebCGMNode.setAppStructureAttr().

Exceptions on setting

WebCGMException; NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly.

ownerNode of type WebCGMNode, readonly

The Element node this attribute is attached to or null if this attribute is not in use.

Methods

No defined methods.

5.7.9 Interface WebCGMEventListener

The WebCGMEventListener interface is the primary method for handling events. Users register their listener on the WebCGMMetafile node with the addEventListener method.

IDL Definition

```
interface WebCGMEventListener {
  void       handleEvent(in WebCGMEvent evt);
};
```

Example of addEventListener

EXAMPLE:

This simple example shows how to use the addEventListener method of the WebCGMMetafile interface from HTML & ECMAScript. In a WebCGM-DOM enabled browser, execution of this HTML-ECMAScript code should display a picture with two black-framed square figures, and a mouse click anywhere on either of them should result in an alert saying "Event handler installed successfully."

```
<html>
<head>
<title>Example for WebCGMEventListener</title>
<script type="text/ecmascript">

var cgmDoc;

function InstallEventListener() {
  try {
    cgmDoc = document.getElementById("ivx1").getWebCGMDocument();
    cgmDoc.src = 'ex_WebCGM_Event.cgm';
    cgmDoc.addEventListener( "click", handleClick);
  }
  catch(e) {
    alert( "Failed:  " + e.description );
  }
}
function handleClick(evt) {
  alert( "Event handler installed successfully." );
}
</script>
</head>

<body onload="InstallEventListener();">
  <table border="1" rules="cols">
    <tr style="text-align:center;">
      <th>Example, WebCGMEventListener Interface</th> </tr>
    <tr>
      <td><object id="ivx1" type="image/cgm;Version=4;ProfileId=WebCGM"
                  width="480" height="360"></object>  </td>  </tr>
  </table>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

Attributes

No defined attributes.

Methods

handleEvent

This method is called whenever an event occurs of the type for which the WebCGMEventListener interface was registered.

Parameters

evt of type WebCGMEvent

The WebCGMEvent containing contextual information about the event.

Return value

No return value.

Exceptions

No exceptions.

5.7.10 Interface `WebCGMEvent`

The WebCGMEvent interface is used to provide contextual information about an event to the handler processing the event.

There exists three levels of interactivity in WebCGM:

- User-initiated actions such as a mouse click can be captured by the host environment and execute scripts.
- The user can initiate hyperlinks to Web pages or other WebCGM illustrations.
- User agent, users are able to zoom into and pan around WebCGM content.

This section also describes how a user agent processes the three different levels of interactivity.

When a mouse event occurs, the WebCGM user agent determines the target object of the mouse event. For the purposes of this discussion, "object" means Application Structure (APS). The target object is the topmost object whose *interactive region* is under the mouse at the time of the event. (Note that the definition of interactive region excludes objects that are fully transparent due to the setting of their graphical attributes.)

An application structure of type 'grnode' or 'layer' cannot be a target of a mouse event. Instead, if the mouse pointer was over a 'grnode' when the event occurred; its closest ancestor object of type 'grobject', 'para' or 'subpara' will be designated as the target element. When an object is not displayed (i.e., 'visibility' attribute is set to off) or made non-interactive (i.e., 'interactivity' attribute is set to off), that object cannot be the target of mouse events.

The event is either initially dispatched to the Metafile, or else not dispatched, depending on the following:

- If there are no graphics objects whose *interactive region* is under the mouse (i.e., there is no target object), the event is not dispatched.
- Otherwise, the Metafile is checked to see if it has an appropriate event handler. The event is dispatched to the event handler if an appropriate one is found.
- Otherwise, the event is discarded.

See also the descriptions of grobject, grnode, para or subpara, for related specifics.

The processing order for user interface events is as follows:

- Event handlers assigned to a WebCGMMetafile get the event first via the potential event bubbling. If none of the activation event handlers take an explicit action (by invoking the preventDefault() WebCGM DOM method) to prevent further processing of the given event, then the event is passed on for:
- Cursor change, screentip and hyperlink processing. If a hyperlink is invoked in response to a user interface event, the hyperlink typically will disable further activation event processing (e.g., hyperlink to another Web page). If link processing does not disable further processing of the given event, then the event is passed on for:
- Document-wide event processing, such as user agent facilities to allow zooming and panning of a WebCGM document.

Since hyperlinks will in general change the context of a document it is more appropriate to allow explicit handlers to act on an event first and then process the hyperlink. The reverse order cannot guarantee that the script would get executed.
Script writers should be made aware that this specification does not cover user agent event facilities such as zooming, panning or context menus. The mechanism to invoke such functionality will likely be different between vendors. Script writers are encouraged to become aware of those differences and thus, write highly interoperable WebCGM scripts.

IDL Definition

```
interface WebCGMEvent {
  readonly attribute WebCGMString     type;
  readonly attribute WebCGMNode       target;
  readonly attribute unsigned short   button;
  readonly attribute long             numPressed;
  readonly attribute float            clientX;
  readonly attribute float            clientY;
  readonly attribute boolean          ctrlKey;
  readonly attribute boolean          shiftKey;
  readonly attribute boolean          altKey;
  readonly attribute boolean          metaKey;

  void               preventDefault();
};
```

Basic example

EXAMPLE:

This simple example shows how to use the attributes of the WebCGMEvent interface from HTML & ECMAScript. In a WebCGM-DOM enabled browser, successful execution of this HTML-ECMAScript code should display a picture with two black-framed square figures. A mouse click anywhere on the left black background should report the X/Y location of the click, and a mouseover of the left ellipse should report the mouseover.

```
<html>
<head>
<title>Example for WebCGMEvent</title>
<script type="text/ecmascript">

  var cgmDoc;

  function handleClick(evt) {
    try {
      if( evt.target.apsId == "grobject_rect_1" ) {
          alert( "ClientX = " + evt.clientX + "  ClientY = " + evt.clientY );
      }
    }
    catch(e) {
      alert( e );
    }
  }
  function handleMOver(evt) {
    try {
      if( evt.target.apsId == "grobject_circle_1" ) {
        alert( "You have mouse-over'd grobject_circle_1" );
      }
    }
    catch(e2) {
        alert( e2 );
    }
  }
  function addHandlers() {
    try {
      cgmDoc = document.getElementById("ivx1").getWebCGMDocument();
      cgmDoc.src = 'ex_WebCGM_Event.cgm';
      cgmDoc.addEventListener ("click", handleClick);
      cgmDoc.addEventListener ("mouseover", handleMOver);
    }
    catch(e4) {
        alert( e4 );
```

```
    }
  }
</script>
</head>
<body onload="addHandlers();">
    <table border="1" rules="cols">
        <tr style="text-align:center;">
            <th>Example, WebCGMEvent Interface</th> </tr>
        <tr>
            <td><object id="ivx1" type="image/cgm;Version=4;ProfileId=WebCGM"
                        width="480" height="360"></object> </td> </tr>
    </table>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

Attributes

type of type WebCGMString, readonly

The name of the event (case-insensitive). The name must be an XML name.

target of type WebCGMNode, readonly

Used to indicate the WebCGMNode (Application Structure) to which the event was originally dispatched.

button of type unsigned short, readonly

During mouse events caused by the depression or release of a mouse button, button is used to indicate which mouse button changed state. The values for button range from zero to indicate the left button of the mouse, one to indicate the middle button if present, and two to indicate the right button. For mice configured for left handed use in which the button actions are reversed the values are instead read from right to left.

numPressed of type long, readonly

Indicates the number of times a mouse button has been pressed and released over the same screen location during a user action. The attribute value is 1 when the user begins this action and increments by 1 for each full sequence of pressing and releasing. If the user moves the mouse between the mousedown and mouseup the value will be set to 0, indicating that no click is occurring.

clientX of type float, readonly

The horizontal coordinate at which the event occurred expressed in Normalized VDC.

clientY of type float, readonly

The vertical coordinate at which the event occurred expressed in Normalized VDC.

ctrlKey of type boolean, readonly

Used to indicate whether the 'ctrl' key was depressed during the firing of the event.

shiftKey of type boolean, readonly

Used to indicate whether the 'shift' key was depressed during the firing of the event.

altKey of type boolean, readonly

Used to indicate whether the 'alt' key was depressed during the firing of the event. On some platforms this key may map to an alternative key name.

metaKey of type boolean, readonly

Used to indicate whether the 'meta' key was depressed during the firing of the event. On some platforms this key may map to an alternative key name.

Methods

preventDefault

Calling preventDefault has the effect of canceling the event. Any default action associated with the event will not occur.

Parameters
No parameters.
Return value
No return value.
Exceptions
No exceptions.

WebCGM supports the following types of events:

click The click event occurs when the pointing device button is clicked. A click is defined as a mousedown and mouseup over the same screen location. The sequence of these events is: mousedown, mouseup, click. If multiple clicks occur at the same screen location, the sequence repeats with the detail attribute incrementing with each repetition. The Application Structure (if any) which was under the mouse pointer when clicked is populated in the WebCGMEvent.target property.

mousedown The mousedown event occurs when the pointing device button is pressed. The Application Structure (if any) which was under the mouse pointer when it was pressed down is populated in the WebCGMEvent.target property.

mouseup The mouseup event occurs when the pointing device button is released. The Application Structure (if any) which was under the mouse pointer when it was released is populated in the WebCGMEvent.target property.

mouseover The mouseover event occurs when the pointing device is moved onto an Application Structure. The Application Structure that the mouse pointer moved over is populated in the WebCGMEvent.target property.

mouseout The mouseout event occurs when the pointing device is moved away from an Application Structure. The Application Structure that the mouse pointer moved away from is populated in the WebCGMEvent.target property.

load The load event occurs when the WebCGM DOM implementation finishes loading all content within a WebCGM metafile.

unload The unload event occurs when the WebCGM DOM implementation removes a WebCGM metafile from a window or frame.

Back to top of chapter

---

6. WebCGM Profile

This chapter and its sections are normative, unless otherwise indicated.

Contents

6.1 WebCGM Proforma

The following profile proforma (PPF) defines the WebCGM application profile for CGM files with a comparison to the ISO Model Profile as defined in ISO/IEC 8632:1999. The tables for the ISO Model Profile are duplicated here for reference and are intended to be accurate. In case of discrepancies, the Model Profile in ISO/IEC 8632:1999 shall take precedence. In the PPF, there are references such as 9.5.4.5, 7.5.11, and Annex I, etc. These are references to sections of the CGM:1999 text, from which this proforma is extracted. Other internal PPF references look like T.16.13 and Attachment 26.3, which are references to table entries in the PPF itself.

The Model Profile (in the ISO CGM:1999 text) uses check boxes to indicate whether an item is required, permitted, or prohibited in metafiles conforming to the Model Profile. Authors of cascading profiles should be aware that the Model Profile does not have all three possible choices for each item, and that alternate choices are indicated here in the tables with "No" and the "checked" choice is indicated with "Yes." For example, this rule "Element is: Required Yes; Permitted No;" means that the element is required in all metafiles conforming to WebCGM 2.1 and that authors of cascading profiles could choose to make the element optional (Permitted), but they must not prohibit the element.

6.2 Metafile Rules

Functionality	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile

T.13.1	Same as Model Profile: **No**	
Encodings	Select 1 or more encodings: Binary **Yes**; Clear text **No**; Other: ***the whole metafile may be GZIP compressed. WebCGM interpreters must support GZIP-compressed metafiles.***	Select 1 or more encodings: Binary **Yes**; Clear text **Yes**;
T.13.2	Same as Model Profile: **No**	
Number of pictures	Number of pictures permitted in a metafile: minimum (>=0)? ***1.*** maximum (>=0 or no limit)? ***1.*** Other: None.	Number of pictures permitted in a metafile: minimum (>=0)? ***1.*** maximum (>=0 or no limit)? ***No limit.*** Other: ***None.***
T.13.3	Same as Model Profile: **Yes**	
Empty pictures		Are pictures allowed which have no graphical primitives? (yes/no) ***Yes***. Other: ***None.***
T.13.4	Same as Model Profile: **Yes**	
Metafile size		Any restrictions on metafile size? ***No.*** Other: ***None.***

6.3 Multi-element Rules

Functionality	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.14.1	Same as Model Profile: **No**	

Colour References: 9.5.4.1	Select which rule applies to each metafile (choose 1): Either all colours or none shall be defined. **Yes**; All colours shall be defined. **No**; No colours shall be defined. **No**; Are colour indexes allowed to be redefined within a picture or metafile? (yes/no) ***Yes.*** Any restrictions on the number of distinct colours used within a picture or metafile? (Monochrome metafiles shall use at most two distinct colours.) ***None.*** Are conformance categories defined? (yes/no) ***Yes.*** If yes, specify. ***Monochrome and colour.*** Other: ***Greyscale is considered to be a special class of colour.***	Select which rule applies to each metafile (choose 1): Either all colours or none shall be defined. ***Yes***; All colours shall be defined. ***No***; No colours shall be defined. ***No***; Are colour indexes allowed to be redefined within a picture or metafile? (yes/no) ***No.*** Any restrictions on the number of distinct colours used within a picture or metafile? (Monochrome metafiles shall use at most two distinct colours.) ***None.*** Are conformance categories defined? (yes/no) ***Yes.*** If yes, specify. ***3 categories: monochrome, greyscale, and colour***. Other: ***None.***
T.14.2	Same as Model Profile: **Yes**	
Line primitives - geometric degeneracies References: 9.5.4.3		Geometric degeneracies are: Permitted **Yes**; Prohibited **No**; If permitted, graphical meaning of the degeneracy: ***A line primitive element, whose entire locus is a single point, denotes a graphical dot which is a filled circle, with diameter equal to the current line width and colour equal to the current line colour.*** Other: ***None.***
T.14.3	Same as Model Profile: **Yes**	
Filled area primitives - geometric degeneracies References: 9.5.4.4		Geometric degeneracies are: Permitted **Yes**; Prohibited **No**; If permitted, graphical meaning of the degeneracy: ***A filled-area primitive element, whose entire locus is either a single point or a line has the following meaning:*** ***- If the locus of a filled-area primitive is a single point, then the meaning is a dot (which is a filled circle).*** ***- If the locus of a filled-area primitive is a non-degenerate line segment, then the meaning is a line.*** ***The dot or line is displayed with the fill colour if EDGE VISIBILITY is 'off', unless INTERIOR STYLE is 'empty', in which case it is not rendered. If EDGE VISIBILITY is 'on', the interior treatment is the dot or line displayed in the fill colour, and then a dot or line superimposed with the current edge attributes.*** Other: ***None.***

T.14.4	Same as Model Profile: **No**	
Graphical text strings References: 9.5.4.5	Minimum string length (bytes): ***0*** Maximum string length (bytes): ***254*** Any restrictions on the use of ISO/IEC 2022 switching controls? ***The C0 character NUL (code value) is permitted and has no effect. String parameters of graphical text shall contain no control character (7/8 bit codes: 1-31 and 128-159). ISO/IEC 2022 switching is not allowed in graphical text. A valid WebCGM metafile may use for graphical text only the character encodings (CGM "character sets"): the collection of four character encodings which comprise ISO Latin1 and Symbol (see CHARACTER SET LIST); Unicode UTF-8; and UTF-16.*** Other: ***None.*** Note. According to the Binary Encoding of CGM:1999, strings of multi-byte Unicode text are "big-endian", like the rest of the binary metafile.	Minimum string length (bytes): 0. Maximum string length (bytes): ***254.*** Any restrictions on the use of ISO/IEC 2022 switching controls? ***C0 control codes (except NUL and ISO/IEC 2022 switching) are prohibited.*** ***Any character set used in the metafile which is accessed by ISO/IEC 2022 switching techniques shall be in the Character Set List (defined in this profile).*** Other: ***None.***
T.14.5	Same as Model Profile: **No**	
Non-graphical text strings References: 9.5.4.6	Maximum string length (bytes): for type SF: ***254*** for type SF within type D: ***1024*** Format effectors and ESC: Permitted **Yes**; Prohibited **No**; Note: according to CGM:1999, the format effectors are NUL, CR, LF, BS, HT, VT, and FF. Other C0 control codes (except NUL and ISO/IEC 2022 switching) are prohibited. Any limits on the set of acceptable character encodings (CGM "character sets")? ***The permitted character encodings (CGM "character sets") for non-graphical text are ISO Latin 1 (LHS & RHS), and UNICODE UTF-8, and Unicode UTF-16. Only one of these three shall be used throughout any particular WebCGM metafile instance. According to the CGM standard, the default SF character encoding ("set"), at the beginning of the 'metafile id' parameter of the BEGIN METAFILE element is ISO Latin 1. If the metafile is to use UTF-8 for SF parameters, then the following 4-octet ISO 2022 sequence shall occur as the first 4 octets of the 'metafile id' parameter:*** **ESC 2/5 2/15 4/9** **If the metafile is to use UTF-16 for SF parameters, then the following 4-octet ISO 2022 sequence shall occur as**	Maximum string length (bytes): for type SF: ***254.*** for type SF within type D: ***1024.*** Format effectors and ESC: Permitted **Yes**; Prohibited **No**; Other C0 control codes (except NUL and ISO/IEC 2022 switching) are prohibited. Any limits on the set of acceptable character sets? ***The permitted character sets are ISO 8859-1 LHS No.1 and ISO 8859-1 RHS No.1.*** Any restrictions on the use of ISO/IEC 2022 switching controls? ***Any character set used in the metafile which is accessed by ISO/IEC 2022 switching techniques shall be in the character set list (defined in this profile).*** Other: ***None.***

	the first 4 octets of the 'metafile id' parameter: **ESC 2/5 2/15 4/12** **Otherwise, the use of ISO 2022 switching is prohibited in non-graphical text string.** ***NOTE: Section 6.3.4.5 of CGM:1999 allows the switching to UTF-8 (variable length multi-byte), and allows the use of 8-bit access to the ISO Latin 1 set.*** Any restrictions on the use of ISO/IEC 2022 switching controls? ***Any character encodings (CGM "character sets") used in the metafile which is accessed by ISO/IEC 2022 switching techniques shall be in the character set list (defined in this profile).*** Other: ***See 3.1.1 for additional restrictions to the character repertoire for those WebCGM non-graphical strings which may be part of the WebCGM IRI fragment.*** Note. According to the Binary Encoding of CGM:1999, strings of multi-byte Unicode text are "big-endian", like the rest of the binary metafile.	
T.14.6	Same as Model Profile: **Yes**	
Data record strings References: 9.5.4.7		Maximum string length (bytes) or state "no limit": ***32767.*** SDR-coding techniques must be used (see annex C.2.2). Other: ***None.***

6.4 Delimiter Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.15.0	Same as Model Profile: **Yes**	
no-op [v1] References: Part 3, 8.2		Element is: Required **No**; Permitted **Yes**; The parameter value of this element is encoding dependent. This element is applicable only to binary encoding. It shall be included in the profile only if binary encoding is permitted or required. If binary encoding is permitted, is the element Required ***No***; Permitted ***Yes***; If permitted, are there any restrictions on the Parameter value? ***None***. Other: ***None.***

T.15.1	Same as Model Profile: **Yes**	
BEGIN METAFILE END METAFILE [v1] References: 7.2.1 7.2.2 9.5.4.6 T.14.5		Element is: Required **Yes**; The *metafile identifier* parameter shall follow the rules for non-graphical text, clause 9.5.4.6 and T.14.5. Other: ***None.***
T.15.2	Same as Model Profile: **No**	
BEGIN PICTURE BEGIN PICTURE BODY END PICTURE [v1] References: 7.2.3 7.2.4 7.2.5 9.5.4.6 T.14.5	Element is: Required **Yes**; The *picture identifier* shall follow the rules for non-graphical text, clause 9.5.4.6 and T.14.5. Number of occurrences of these elements allowed in the metafile: ***1.*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *picture identifier* shall follow the rules for non-graphical text, clause 9.5.4.6 and T.14.5. Number of occurrences of these elements allowed in the metafile: ***No limit.*** Other: ***None.***
T.15.3	Same as Model Profile: **No**	

BEGIN SEGMENT END SEGMENT [v2] References: 7.2.6 7.2.7	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Maximum number of simultaneously defined segments (both global and local) at any point in the metafile: Any limits on the number of elements or restrictions on which elements compose a segment? Is there any meaning given to the *segment identifier* parameter? (yes/no) If yes, specify. (Meaning shall have no graphical effect.) Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneously defined segments (both global and local) at any point in the metafile: ***1024.*** Any limits on the number of elements or restrictions on which elements compose a segment? ***None.*** Is there any meaning given to the *segment identifier* parameter? (yes/no) ***No.*** If yes, specify. (Meaning shall have no graphical effect). Other: ***When global segments are specified in the Metafile Descriptor, all global segment definitions shall follow all other Metafile Descriptor elements. When segments are specified in the Picture Descriptor, all such segment definitions shall follow all other Picture Descriptor elements.***
T.15.4	Same as Model Profile: **No**	
BEGIN FIGURE END FIGURE [v2] References: 7.2.8 7.2.9	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Limits on the number of elements or restrictions on which elements comprise a figure definition: ***Maximum number of elements = 1024. No restrictions on which eligible elements may be included.*** Other: ***Note that the 1024 element limit applies to the maximum number of graphical primitive elements. Eligible elements of classes other than graphical primitives that are included within the CLOSED FIGURE, e.g., primitive attribute elements, do not count against the 1024 limit.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Limits on the number of elements or restrictions on which elements comprise a figure definition: ***Maximum number of elements = 128. No restrictions on which eligible elements may be included.*** Other: ***None.***
T.15.5	Same as Model Profile: **No**	
BEGIN PROTECTION REGION END PROTECTION REGION [v3] References: 7.2.10 7.2.11	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneously defined protection regions: ***1***. Maximum number of elements within each protection region: ***128***. Is there any meaning to the region index parameter other than as a unique identifier for each protection region? (yes/no) ***No***. If yes, specify. (Meaning shall have no graphical effect). Other: ***Region index is restricted to the value "1".***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneously defined protection regions: ***32.*** Maximum number of elements within each protection region: ***128.*** Is there any meaning to the region index parameter other than as a unique identifier for each protection region? (yes/no) No. If yes, specify. (Meaning shall have no graphical effect). Other: ***None.***
T.15.6	Same as Model Profile: **Yes**	

BEGIN COMPOUND LINE END COMPOUND LINE [v3] References: 7.2.12 7.2.13		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Limits on the number of elements and identity of elements comprising a path definition: ***Maximum number of elements is 128. No restrictions on which eligible elements may be included.*** Other: ***None.***
T.15.7	Same as Model Profile: **Yes**	
BEGIN COMPOUND TEXT PATH END COMPOUND TEXT PATH [v3] References: 7.2.14 7.2.15		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Limits on the number and identity of elements comprising a path definition: ***Maximum number of elements is 128. No restrictions on which eligible elements may be included.*** Other: ***None.***
T.15.8	Same as Model Profile: **No**	
BEGIN TILE ARRAY END TILE ARRAY [v3] References: 7.2.16 7.2.17	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of tiles in path direction: ***64*** Maximum number of tiles in line direction: ***64*** Maximum number of cells/tile in path direction: ***4096*** Maximum number of cells/tile in line direction: ***4096*** Limits on pel path: ***shall be 0.*** Limits on line progression: ***None.*** Limits on image offset: ***None.*** Other: ***Two types of raster images are allowed. A single (non-tiled) image has a maximum of 1,073,741,824 (1 giga, 32768**2) cells. A non-tiled image may exceed the 4096 cells/tile restriction specified for the tiled images. Tiled raster images are limited 64X64 tiles which are a maximum of 4096X4096 cells each. These tiled images***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of tiles in path direction: ***16.*** Maximum number of tiles in line direction: ***16.*** Maximum number of cells/tile in path direction: ***1024.*** Maximum number of cells/tile in line direction: ***1024.*** Limits on pel path: ***None.*** Limits on line progression: ***None.*** Limits on image offset: ***None.*** Other: ***None.***

	are limited to a total of 1,073,741,824 (1 giga, 32768**2) cells (adjustment of maximum number of tiles and cells per tile are necessary to meet this requirement).	
T.15.9	Same as Model Profile: **No**	
BEGIN APPLICATION STRUCTURE BEGIN APPLICATION STRUCTURE BODY END APPLICATION STRUCTURE [v4] References: 7.2.18 7.2.19 7.2.20	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Limits on the maximum number of defined structures within a picture:***None.*** Limits on the number and identity of elements comprising a structure: ***None.*** Is there any meaning to the Application Structure (APS) identifier parameter? yes/no ***No. No assigned meaning beyond being a unique identifier for the application structure.*** If yes, specify. ***n/a*** Is the inheritance flag parameter restricted? Yes/no: **Yes**. **The value of the inheritance flag is restricted to a value corresponding to "statelist"**. Other: **The value of the structure type parameter must be chosen from the list of valid structure types listed in the Section 3.2.1. Structures are placed in the metafile according to the EBNF content model fragments in Section 3.2.1 and its subsections. The character repertoire of the APS id parameter is identical to that of the `objid` production as defined in Section 3.1.1.3.**	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Limits on the maximum number of defined structures within a picture: ***None.*** Limits on the number and identity of elements comprising a structure: ***None.*** Is there any meaning to the application structure identifier parameter? yes/no ***No. No assigned meaning beyond being a unique identifier for the application structure.*** If yes, specify. Is the inheritance flag parameter restricted? Yes/no: ***No*** Other: ***None.***

6.5 Metafile Descriptor Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.16.1	Same as Model Profile: **Yes**	
METAFILE VERSION [v1] References: 7.3.1		Element is: Required **Yes**; Metafile versions permitted by this profile:***1, 2, 3, 4*** Other: ***None.***
T.16.2	Same as Model Profile: **No**	

METAFILE DESCRIPTION [v1] References: 7.3.2 9.5.2.1 9.5.2.2 9.5.4.6 T.14.1 T.14.5	Element is: Required **Yes**; The *description* parameter shall follow the rules for non-graphical text, clause 9.5.4.6 and T.14.5. The substring within the SF parameter shall be of the form: "keyword:item", where the double quotes are part of the substring. Maximum number of occurrences of this element? ***1*** Profile identification (use keyword, "ProfileId:"): ***"ProfileId: WebCGM".*** Profile edition (use keyword, "ProfileEd:"): ***Refers to the approved version and revision of the specification that applies for this graphic. The Item associated with the keyword ProfileEd shall be n.m. For this WebCGM Edition: "ProfileEd:2.1".*** Additional information content: Metafile colour conformance class, source, and date items shall be encoded as substrings of the *description* parameter using the keywords: "ColourClass:", "Source:", and "Date:", respectively. ColourClass: Required **Yes**; • Content: ***One of "ColourClass:monochrome" or "ColourClass:colour".*** Source? Required **No**; Permitted **Yes**; • Content: "***Source:supplier"*** Date? Required **No**; Permitted **Yes**; • Content: ***"Date:yyyymmdd"*** Other: ***Parameter strings are considered case insensitive.***	Element is: Required **Yes**; The *description* parameter shall follow the rules for non-graphical text, clause 9.5.4.6 and T.14.5. The substring within the SF parameter shall be of the form: "keyword:item", where the double quotes are part of the substring. Maximum number of occurrences of this element? ***Unlimited.*** Profile identification (use keyword, "ProfileId:"): ***"ProfileId: Model-Profile".*** Profile edition (use keyword, "ProfileEd:"): ***"ProfileEd:2".*** If the profile edition is not given, then the edition defaults to 1. Additional information content: Metafile colour conformance class, source, and date items shall be encoded as substrings of the *description* parameter using the keywords: "ColourClass:", "Source:", and "Date:", respectively. ColourClass: Required **Yes**; Permitted **No**; • Content: ***(One of: colour, greyscale, or monochrome.)*** Source? Required **Yes**; Permitted **No**; • Content: ***(Vendor, product, and version).*** Date? Required **Yes**; Permitted **No**; • Content shall be date of metafile generation. ***The form and content shall be in accordance with ISO 8601:1988.*** Other: ***None.***
T.16.3	Same as Model Profile: **Yes**	
VDC TYPE [v1] References: 7.3.3		Element is: Required **No**; Permitted **Yes**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.16.4	Same as Model Profile: **No**	

INTEGER PRECISION [v1] References: 7.3.4	Element is: Required **No**; Permitted **Yes**; The parameter value of this element is encoding dependent. If binary encoding is permitted, are there any restrictions on the parameter value? ***16, or 32***. Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***n/a*** Other: ***n/a***	Element is: Required **No**; Permitted **Yes**; The parameter value of this element is encoding dependent. If binary encoding is permitted, are there any restrictions on the parameter value? ***8, 16, or 32*.** Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***[-256,255], [-32767,32767], [-32768,32767] or [-2147483648,2147483647]*** Other: ***None.***
T.16.5	Same as Model Profile: **Yes**	
REAL PRECISION [v1] References: 7.3.5		Element is: Required **No**; Permitted **Yes**; The parameter value of this element is encoding dependent. If binary encoding is permitted, are there any restrictions on the parameter value? ***(1, 16, 16) or (0, 9, 23)***. Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***-32767,+32767, 4; or -32768, +32767, 10; or -3.4028235E38, +3.4028235E38, 8*** ***Note: The latter two values are the closest approximation, in base 10 clear text, to the REAL PRECISION values allowed in binary encoded CGMs.*** Other: ***None.***
T.16.6	Same as Model Profile: **No**	
INDEX PRECISION [v1] References: 7.3.6	Element is: Required **No**; Permitted **Yes**; The parameter value of this element is encoding dependent. If binary encoding is permitted, if permitted, are there any restrictions on the parameter value? ***16.*** Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***n/a*** Other: ***n/a***	Element is: Required **No**; Permitted **Yes**; The parameter value of this element is encoding dependent. If binary encoding is permitted, If permitted, are there any restrictions on the parametervalue? ***8, 16, or 32***. Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***[0, 127], [-256, 255], [-32767, 32767], [-32768, 32767], or [-2147483648, 2147483647]*** Other: ***None.***
T.16.7	Same as Model Profile: **Yes**	

COLOUR PRECISION [v1] References: 7.3.7		Element is: Required **No**; Permitted **Yes**; The parameter value of this element is encoding dependent. If binary encoding is permitted, are there any restrictions on the parameter value? ***8 or 16.*** Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***255 or 65535.*** Other: ***None.***
T.16.8	Same as Model Profile: **Yes**	
COLOUR INDEX PRECISION [v1] References: 7.3.8		Element is: Required **No**; Permitted **Yes**; The parameter value of this element is encoding dependent. If binary encoding is permitted, are there any restrictions on the parameter value? ***8 or 16***. Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***127, 255, 32767***. Other: ***None.***
T.16.9	Same as Model Profile: **No**	
MAXIMUM COLOUR INDEX [v1] References: 7.3.9	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Is this element required to be a least upper bound? (yes/no) ***No.*** Any restrictions on the parameter values? • ***0-1 for monochrome metafiles.*** • ***0-255 for colour metafiles.*** Other: ***Greyscale is considered a special case of colour.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Is this element required to be a least upper bound? (yes/no) ***No.*** Any restrictions on the parameter values? • ***0-1 for monochrome metafiles.*** • ***0-63 for greyscale metafiles.*** • ***0-255 for colour metafiles.*** Other: ***None.***
T.16.10	Same as Model Profile: **Yes**	

COLOUR VALUE EXTENT [v1] References: 7.3.10		Element is: Required **No**; Permitted **Yes**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.16.11	Same as Model Profile: **Yes**	
METAFILE ELEMENT LIST [v1] References: 7.3.11		Element is: Required **Yes**; Other: ***None.***
T.16.12	Same as Model Profile: **Yes**	
METAFILE DEFAULTS REPLACEMENT [v1] References: 7.3.12		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Is each occurrence of the MDR restricted to defining just one default? (yes/no) ***No.*** Additional restrictions may be specified in parts 3 and 4 of ISO/IEC 8632. Other: ***None.***
T.16.13	Same as Model Profile: **No**	
FONT LIST [v1] References: 7.3.13 annex I	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; This element is required for all metafiles containing graphical text. Maximum number of fonts in the list: ***256*** All font indexes referenced in the metafile, including the default (nominally index 1) shall be defined in the FONT LIST element, with font name construction consistent with the rules of ISO/IEC 9541. List of ***recommended fonts:*** • ***Times-Roman*** • ***Times-Bold*** • ***Times-Italic*** • ***Times-BoldItalic*** • ***Helvetica*** • ***Helvetica-Bold*** • ***Helvetica-Oblique*** • ***Helvetica-BoldOblique***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; This element is required for all metafiles containing graphical text. Maximum number of fonts in the list: ***64.*** All font indexes referenced in the metafile, including the default (nominally index 1) shall be defined in the FONT LIST element, with font name construction consistent with the rules of ISO/IEC 9541. List of permitted fonts: • ***Times-Roman*** • ***Times-Bold*** • ***Times-Italic*** • ***Times-BoldItalic*** • ***Helvetica*** • ***Helvetica-Bold*** • ***Helvetica-Oblique*** • ***Helvetica-BoldOblique***

	• ***Courier*** • ***Courier-Bold*** • ***Courier-Oblique*** • ***Courier-BoldOblique*** • ***Symbol*** NOTE - Theses font names are trademarked and some are proprietary and copyrighted. Times and Helvetica are registered trademarks of Allied Corporation, the owner of the copyright on the fonts of those names. Metric equivalents of the named fonts may be substituted by interpreters. Times is a serif font. Helvetica is a sans-serif font. Courier is a monospaced, serif font. The association of character code to glyph which shall be used for each of the fonts and the metrics of the named fonts are contained in clause I.2, annex I of CGM:1999. Other: ***The list of recommended fonts is intended to be a list of required minimum interpreter font capability and a recommended maximum font capability for generators. If other fonts are used, the FONT PROPERTIES and RESTRICTED TEXT elements are required. Font names are considered case insensitive.***	• ***Courier*** • ***Courier-Bold*** • ***Courier-Oblique*** • ***Courier-BoldOblique*** • ***Symbol*** NOTE - Theses font names are trademarked and some are proprietary and copyrighted. Times and Helvetica are registered trademarks of Allied Corporation, the owner of the copyright on the fonts of those names. Metric equivalents of the named fonts may be substituted by interpreters. Times is a serif font. Helvetica is a sans-serif font. Courier is a monospaced, serif font. The association of character code to glyph which shall be used for each of the fonts and the metrics of the named fonts are contained in clause I.2, annex I. Other: ***None.***
T.16.14	Same as Model Profile: **No**	
CHARACTER SET LIST [v1] References: 7.3.14	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; This element is required for all metafiles containing graphical text. *Note. The terminology "character set", used in the original ISO CGM:1987 specification and preserved through CGM:1999, is considered inaccurate by contemporary standards. The current correct terminology is character encoding, defined in the [CHARMOD] standard.* Maximum limit for the number of character sets in the character set list: ***6.*** Allowable character sets: ***"94-character G-set", 4/2 (ISO 8859-1 LH);*** ***"96-character G-set", 4/1 (ISO 8859-1 RH);*** ***"94-character G-set", 2/10 3/10 (Symbol LH);*** ***"94-character G-set", 2/6 3/10 (Symbol RH);***. ***"complete code", 2/15 4/9 (UTF-8)*** ***"complete code", 2/15 4/12 (UTF-16)*** *Note. The tails for UTF-8 and UTF-16 differ from their WebCGM 1.0 values. 2.1 metafiles shall not use the 1.0 forms.* If any of these character sets is of type "complete code", specify the content of the complete code and its associated sequence tail: ***Specified***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; This element is required for all metafiles containing graphical text. Maximum limit for the number of character sets in the character set list: ***4.*** Character sets shall be selected from the ISO Registry of Character Sets. This list may be extended by adding profile-defined character sets. List character sets: ***"94-character G-set", 4/2 (ISO 8859-1 LH);*** ***"96-character G-set", 4/1 (ISO 8859-1 RH);*** ***"94-character G-set", 2/10 3/10 (Symbol LH);*** ***"94-character G-set", 2/6 3/10 (Symbol RH).*** If any of these character sets is of type "complete code", specify the content of the complete code and its associated sequence tail: ***Not applicable.*** Other: ***None.***

	Other: ***None.***	
T.16.15	Same as Model Profile: **No**	
CHARACTER CODING ANNOUNCER [v1] References: 7.3.15	Element is: Required **Yes**; Any restrictions on the parameter values? ***Value shall be 'basic 8-bit'.*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***Values shall be 'basic 7-bit' and 'basic 8-bit'.*** Other: ***None.***
T.16.16	Same as Model Profile: **No**	
NAME PRECISION [v2] References: 7.3.16 Part 3, 8.3 Part 4, 7.2	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; The parameter value of this element is coding dependent. If binary encoding is permitted, are there any restrictions on the parameter value? ***n/a***. Other: ***n/a***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***n/a***. Other: ***n/a.***	Element is: Required **No**; Permitted **Yes**; The parameter value of this element is coding dependent. If binary encoding is permitted, are there any restrictions on the parameter value? ***8 or 16***. Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***127, 255, 32767***. Other: ***None.***
T.16.17	Same as Model Profile: **Yes**	
MAXIMUM VDC EXTENT [v2] References: 7.3.17		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***None.*** Other: ***None.***
T.16.18	Same as Model Profile: **No**	
SEGMENT PRIORITY EXTENT [v2] References: 7.3.18	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the parameter values***.*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***None.*** Other: ***None.***

T.16.19	Same as Model Profile: **No**	
COLOUR MODEL [v3] References: 7.3.19	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of colour models? ***Shall be 1, 6, 7, or 8.*** Other: ***Values 6, 7, and 8 are the registered values for RGB-alpha, sRGB, and sRGB-alpha.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of colour models? ***None.*** Other: ***None.***
T.16.20	Same as Model Profile: **No**	
COLOUR CALIBRATION [v3] References: 7.3.20	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Calibration selection values permitted in accordance with the permitted model(s): If CYMK is permitted, minimum number of grid locations: Any restrictions on the number of colour lookup table entries, n? Any restrictions on the number of grid locations, m? If CYMK is permitted, algorithms for interpolation between grid locations? Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Calibration selection values permitted in accordance with the permitted model(s): ***Values 1..6, 9.*** If CYMK is permitted, minimum number of grid locations: ***3.*** Any restrictions on the number of colour lookup table entries, n? ***None.*** Any restrictions on the number of grid locations, m? ***None.*** If CYMK is permitted, algorithms for interpolation between grid locations? ***None.*** Other: ***None.***
T.16.21	Same as Model Profile: **No**	
FONT PROPERTIES [v3] References: 7.3.21	Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***The required parameters, when used, are INDEX, FONT FAMILY, POSTURE, WEIGHT, PROPORTIONATE WIDTH, DESIGN GROUP and STRUCTURE.*** Other: ***This element is required when a font is used that is not in the list of recommended fonts specified in the FONT LIST element. Parameter values of type SF are considered to be case insensitive***	Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***All defined index and enumerated values of all parameters shall be permitted.*** Other: ***None.***
T.16.22	Same as Model Profile: **No**	

GLYPH MAPPING [v3] References: 7.3.22	Required **No**; Permitted **No**; Prohibited **Yes**; Subset of AFII registered glyphs which may be referenced: Maximum number of glyphs which may be defined: Other: ***None.***	Required **No**; Permitted **Yes**; Prohibited **No**; Subset of AFII registered glyphs which may be referenced: ***None.*** Maximum number of glyphs which may be defined: ***8192.*** Other: ***None.***
T.16.23	Same as Model Profile: **Yes**	
SYMBOL LIBRARY LIST [v3] References: 7.3.23		Required **No**; Permitted **No**; Prohibited **Yes**; Libraries which may be accessed and their encoding rules: Maximum number of libraries which may be accessed: Other: ***NOTE - There are currently no registered symbol libraries.***
T.16.24	Same as Model Profile: **No**	
PICTURE DIRECTORY [v4] References: 7.3.24 9.5.4.6 T.14.5	Required **No**; Permitted **No**; Prohibited **Yes**; Follow the rules for non-graphical text strings for picture identifier parameter, clause 9.5.4.6 and T.14.5. If present, shall PICTURE DIRECTORY elements be complete, i.e., have an entry for every picture in the metafile? (yes/no): ***n/a.*** If "no", describe any special meaning associated with those entries which appear in PICTURE DIRECTORY elements which are incomplete. ***n/a*** Other: ***None.***	Required **No**; Permitted **Yes**; Prohibited **No**; Follow the rules for non-graphical text strings for picture identifier parameter, clause 9.5.4.6 and T.14.5. If present, shall PICTURE DIRECTORY elements be complete, i.e., have an entry for every picture in the metafile? (yes/no) ***Yes***. If "no", describe any special meaning associated with those entries which appear in PICTURE DIRECTORY elements which are incomplete. Other: ***None.***

6.6 Picture Descriptor Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.17.1	Same as Model Profile: **No**	

SCALING MODE [v1] References: 7.4.1	Element: Required **Yes**; Any restrictions on the parameter values? ***SCALING MODE shall be metric.*** Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***If SCALING MODE is metric then the 'metric scale factor' shall be positive.*** Other: ***None.***
T.17.2	Same as Model Profile: **Yes**	
COLOUR SELECTION MODE [v1][v2] References: 7.4.2		Element: Required **No**; Permitted **Yes**; Any restrictions on the parameter values? ***None.*** Other: ***None.***
T.17.3	Same as Model Profile: **Yes**	
LINE WIDTH SPECIFICATION MODE [v1][v2] References: 7.4.3		Element: Required **No**; Permitted **Yes**; Any restrictions on the parameter values? ***None.*** Other: ***None.***
T.17.4	Same as Model Profile: **Yes**	
MARKER SIZE SPECIFICATION MODE [v1][v2] References: 7.4.4		Element: Required **No**; Permitted **Yes**; Any restrictions on the parameter values? ***None.*** Other: ***None.***
T.17.5	Same as Model Profile: **Yes**	

EDGE WIDTH SPECIFICATION MODE [v1][v2] References: 7.4.5		Element: Required **No**; Permitted **Yes**; Any restrictions on the parameter values? ***None.*** Other: ***None.***
T.17.6	Same as Model Profile: **Yes**	
VDC EXTENT [v1] References: 7.4.6		Element: Required **No**; Permitted **Yes**; Limits on the sense and orientation of the VDC space: ***None.*** Is zero-area VDC extent permitted? (yes/no) ***No.*** If yes, specify its meaning. Other: ***None.***
T.17.7	Same as Model Profile: **Yes**	
BACKGROUND COLOUR [v1] References: 7.4.7 9.5.4.1 T.14.1		Element: Required **No**; Permitted **Yes**; The *colour value* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Other: ***None.***
T.17.8	Same as Model Profile: **Yes**	
DEVICE VIEWPORT [v2] References: 7.4.8		Element: Required **No**; Permitted **No**; Prohibited **Yes**; Interaction of this element with environmental presentation directives: Meaning of this element if the specified value is inconsistent with the presentation device: Other: ***NOTE - This element is prohibited due to its device dependence.***

T.17.9	Same as Model Profile: **Yes**	
DEVICE VIEWPORT SPECIFICATION MODE [v2] References: 7.4.9		Element: Required **No**; Permitted **No** Prohibited **Yes**; Set of legal values: Other: ***NOTE - This element is prohibited due to its device dependence.***
T.17.10	Same as Model Profile: **Yes**	
DEVICE VIEWPORT MAPPING [v2] References: 7.4.10		Element: Required **No**; Permitted **No**; Prohibited **Yes**; Set of legal values: Other: ***NOTE - This element is prohibited due to its device dependence.***
T.17.11	Same as Model Profile: **No**	
LINE REP-RESENTATION [v2] References: 7.4.11 9.5.2.6 9.5.4.2 T.20.1	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Maximum number of simultaneous bundle definitions: Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneous bundle definitions: ***20.*** Other: ***None.***
T.17.12	Same as Model Profile: **No**	

MARKER REP-RESENTATION [v2] References: 7.4.12 9.5.2.6 9.5.4.2 T.20.5	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Maximum number of simultaneous bundle definitions: Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneous bundle definitions: ***20.*** Other: ***None.***
T.17.13	Same as Model Profile: **No**	
TEXT REP-RESENTATION [v2] References: 7.4.13 9.5.2.6 9.5.4.2 T.20.9	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Maximum number of simultaneous bundle definitions: Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneous bundle definitions: ***20.*** Other: ***None.***
T.17.14	Same as Model Profile: **No**	
FILL REP-RESENTATION [v2] References: 7.4.14 9.5.2.6 9.5.4.2 T.20.21	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Maximum number of simultaneous bundle definitions: Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneous bundle definitions: ***20.*** Other: ***None.***
T.17.15	Same as Model Profile: **No**	

EDGE REP-RESENTATION [v2] References: 7.4.15 9.5.2.6 9.5.4.2 T.20.26	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Maximum number of simultaneous bundle definitions: Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneous bundle definitions: ***20.*** Other: ***None.***
T.17.16	Same as Model Profile: **Yes**	
INTERIOR STYLE SPECIFICATION MODE [v3] References: 7.4.16		Element: Required **No**; Permitted **Yes**; Any restriction on the parameter value? ***None.*** Other: ***None.***
T.17.17	Same as Model Profile: **Yes**	
LINE AND EDGE TYPE DEFINITION [v3] References: 7.4.17		Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any limits on the number of definitions? ***Maximum of 32 line types shall be specified simultaneously.*** Any limits on the number of elements in a given definition? ***Number of values in the dash gap list shall not exceed 8.*** Any restrictions on the dash cycle repeat length? ***None.*** Any restrictions on complexity of definition to prevent degeneracies? ***None.*** Other: ***None.***
T.17.18	Same as Model Profile: **Yes**	

HATCH STYLE DEFINITION [v3] References: 7.4.18		Element: Required **No**; Permitted **Yes**; Prohibited **No**; Limit on the number of hatch styles? ***Maximum of 32 hatch styles shall be specified simultaneously.*** Limit on the number of gaps in a given definition? ***Number of entries in the gap width list shall not exceed 8.*** Any limits on duty cycle length? ***None.*** Any restrictions on complexity of definition to prevent degeneracies? ***None.*** Any restrictions on the style indicator? ***None.*** Other: ***None.***
T.17.19	Same as Model Profile: **No**	
GEOMETRIC PATTERN DEFINITION [v3] References: 7.4.19	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Any limits on the number of geometric patterns defined? Any limits on the classes of primitives? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any limits on the number of geometric patterns defined? ***The maximum number of geometric patterns is 64.*** NOTE - The number of geometric patterns cannot exceed the number of segments. Any limits on the classes of primitives? ***None.*** Other: ***None.*** NOTE - The number of geometric patterns cannot exceed the number of segments.
T.17.20	Same as Model Profile: **No**	
APPLICATION STRUCTURE DIRECTORY [v4] References: 7.4.20 9.5.4.6 T.14.5	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Follow rules for non-graphical text strings *for application structure identifier* parameter, clause 9.5.4.6 and T.14.5. Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Follows rules for non-graphical text strings *for application structure identifier* parameter, clause 9.5.4.6 and T.14.5. If present, shall APPLICATION STRUCTURE DIRECTORY elements be complete, i.e., have an entry for every application structure in the picture? (yes/no) ***Yes***. If "no", describe any special meaning associated with those entries which appear in APPLICATION STRUCTURE DIRECTORY elements which are incomplete. Other: ***None.***

6.7 Control Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile

T.18.1	Same as Model Profile: **Yes**	
VDC INTEGER PRECISION [v1] References: 7.5.1 Part 3, 8.5 Part 4, 7.4		Element is: Required **No**; Permitted **Yes**; The parameter values of this element are encoding dependent. If binary encoding is permitted, are there any restrictions on the Parameter value? ***16 or 32.*** Other: ***None***. If clear text encoding is permitted, are there any restrictions on the parameter value? ***[-32767,32767], [-32768, 32767], or [-2147483648,2147483647].*** Other: ***None.***
T.18.2	Same as Model Profile: **Yes**	
VDC REAL PRECISION [v1] References: 7.5.2 Part 3, 8.5 Part 4, 7.4		Element is: Required **No**; Permitted **Yes**; The parameter values of this element are encoding dependent. If binary encoding is permitted, are there any restrictions on the Parameter value? ***(1, 16, 16) or (0, 9, 32)*** Other: ***None***. If clear text encoding is permitted, are there any If clear text encoding is permitted, are there any restrictions on the parameter value? ***0.0, 1.0 , 4; or -32767, 32767, 4; or -32768, 32767, 10; or -3.4028235E38, +3.4028235E38, 8*** ***Note: The latter two values are the closest approximation, in base 10 clear text, to the REAL PRECISION values allowed in binary encoded CGMs.*** Other: ***None.***
T.18.3	Same as Model Profile: **Yes**	
AUXILIARY COLOUR [v1] References: 7.5.3 9.5.4.1 T.14.1 D.4.4.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *auxiliary colour specifier* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Other: ***None.***

T.18.4	Same as Model Profile: **Yes**	
TRANSPARENCY [v1] References: 7.5.4 9.5.7.9 T.14.1	Note: In terms of the drawing model description of section 2.2.2, the conceptual effect of Transparency is as follows. When Transparency is 'on' (default), then for the items affected by Auxiliary Colour and Transparency (inter-dash spaces, etc) the (Pr, Pg, Pb, Pa) in the equations is set to (0,0,0,0) -- transparent black. When Transparency is 'off', then for affected items the (Pr, Pg, Pb, Pa) is set to (r,g,b,a) of the Transparent Color, if RGB-alpha is the colour model. If the colour model is simple RGB, then the (Pr, Pg, Pb, Pa) is set to (r,g,b,a'), where a' is the Esc-45 alpha value in effect at the time, or 1 if there is no such Esc.45.	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restriction on the parameter value? ***None.*** Other: ***None.***
T.18.5	Same as Model Profile: **Yes**	
CLIP RECTANGLE [v1] References: 7.5.5 D.4.4.2		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Meaning of boundary cases for: zero-area: Prohibited. area greater than VDC extent: Clipping shall be done to the intersection of CLIP RECTANGLE and VDC EXTENT. additional cases: None. Other: ***None.***
T.18.6	Same as Model Profile: **Yes**	
CLIP INDICATOR [v1] References: 7.5.6		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.18.7	Same as Model Profile: **No**	
LINE CLIPPING MODE [v2] References: 7.5.7 D.4.4.3	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the parameter value? ***None.*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***

T.18.8	Same as Model Profile: **No**	
MARKER CLIPPING MODE [v2] References: 7.5.8 D.4.4.3	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the parameter value? ***None.*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.18.9	Same as Model Profile: **No**	
EDGE CLIPPING MODE [v2] References: 7.5.9 D.4.4.3	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the parameter value? ***None.*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.18.10	Same as Model Profile: **Yes**	
NEW REGION [v2] References: 7.5.10		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; This element shall be permitted only if BEGIN FIGURE is permitted. Any restrictions on the number of occurrences? ***None.*** Other: ***None.***
T.18.11	Same as Model Profile: **No**	
SAVE PRIMITIVE CONTEXT [v2] References: 7.5.11	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Maximum number of simultaneously saved contexts: Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of simultaneously saved contexts: ***1024.*** Other: ***None.***
T.18.12	Same as Model Profile: **No**	

RESTORE PRIMITIVE CONTEXT [v2] References: 7.5.12	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; This element is permitted only if SAVE PRIMITIVE CONTEXT is permitted. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; This element is permitted only if SAVE PRIMITIVE CONTEXT is permitted. Other: ***None.***
T.18.13	Same as Model Profile: **No**	
PROTECTION REGION INDICATOR [v3] References: 7.5.13	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The values are restricted to: ***off, clip***. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; This element shall be permitted only if BEGIN PROTECTION REGION is permitted. Other: ***None.***
T.18.14	Same as Model Profile: **No**	
GENERALIZED TEXT PATH MODE [v3] References: 7.5.14	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***off, axis-tangential*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.18.15	Same as Model Profile: **No**	
MITRE LIMIT [v3] References: 7.5.15	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***See additional interpreter specifications for mitre limit handling.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.18.16	Same as Model Profile: **Yes**	

TRANSPARENT CELL COLOUR [v3] References: 7.5.16 9.5.4.1 T14.1	"Note: In terms of the drawing model summary of section 2.2.2, the effect of Transparent Cell Colour (TCC) is described as follows. For any cell whose color matches the specified TCC, the (Pr, Pg, Pb, Pa) for that cell in the equations of 2.2.2 is set to (0,0,0,0) -- transparent black. Note that TCC is legacy functionality -- the same thing can be achieved better with RGB-alpha color alone, and mixing TCC and RGB-alpha does not make sense."	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *transparent cell colour specifier* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Any restrictions on the parameter values? ***None.*** Other: ***None.***

6.8 Graphical Primitive Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.19.1	Same as Model Profile: **Yes**	
POLYLINE [v1] References: 7.6.1 T.14.2 D.2.21		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of points or state "no limit": ***4096.*** Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.2	Same as Model Profile: **Yes**	
DISJOINT POLYLINE [v1] References: 7.6.2 T.14.2 D.2.2.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of points or state "no limit": ***4096.*** Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.3	Same as Model Profile: **Yes**	

POLYMARKER [v1] References: 7.6.3		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of points or state "no limit": ***4096.*** Other: ***None.***
T.19.4	Same as Model Profile: **No**	
TEXT [v1] References: 7.6.4 9.5.4.5	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; The *string* parameter shall follow the rules for graphical text, clause 9.5.4.5. Is the 'not final' flag allowed: (yes/no) Other: ***Graphical text shall be represented by the Restricted Text element in this profile.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *string* parameter shall follow the rules for graphical text, clause 9.5.4.5. Is the 'not final' flag allowed: (yes/no) ***Yes.*** Other: ***None.***
T.19.5	Same as Model Profile: **Yes**	
RESTRICTED TEXT [v1] References: 7.6.5 9.5.4.5 T.26.7 D.4.5.2		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *string* parameter shall follow the rules for graphical text, clause 9.5.4.5. Is the 'not final' flag allowed: (yes/no) ***Yes.*** For[v1/2] metafiles, is the realization of RESTRICTED TEXT according to one of the standard or registered values for RESTRICTED TEXT TYPE? (yes/no)***Yes.*** If yes, specify. Boxed-cap, also see T.26.7 For [v3] and [v4] metafiles, RESTRICTED TEXT TYPE shall be used if this element is used. Other: ***None.***
T.19.6	Same as Model Profile: **Yes**	
APPEND TEXT [v1] References: 7.6.6. 9.5.4.5 D.4.5.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *string* parameter shall follow the rules for graphical text, clause 9.5.4.5. Other: ***None.***

T.19.7	Same as Model Profile: **Yes**	
POLYGON [v1] References: 7.6.7 T.14.3 D.2.2.2		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of points: *4096*. Zero-area geometric degeneracies shall be as defined in T.14.3. Other: ***None.***
T.19.8	Same as Model Profile: **Yes**	
POLYGON SET [v1] References: 7.6.8 T.14.3 D.2.2.2		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of points: ***4096.*** Number of polygons in a set? ***No limit.*** Zero-area geometric degeneracies shall be as defined in T.14.3. Other: ***Each individual polygon within a set shall have at least 3 points.***
T.19.9	Same as Model Profile: **No**	
CELL ARRAY [v1] References: 7.6.9 D.4.5.3	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Limit for nx: ***32768*** Limit for ny: ***32768*** Limit for nx*ny:***1,073,741,824 ("1 giga", 32768**2).*** Are rotated and skewed cell arrays allowed? (yes/no) ***No.*** If yes, specify the graphical meaning. Other: ***Zero-area cell arrays are prohibited.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Limit for nx: ***2048.*** Limit for ny: ***2048.*** Limit for nx*ny: ***4194304.*** Are rotated and skewed cell arrays allowed? (yes/no) ***No.*** If yes, specify the graphical meaning. Other: ***Zero-area cell arrays are prohibited.***
T.19.10	Same as Model Profile: **Yes**	

GENERALIZED DRAWING PRIMITIVE [v1] References: 7.6.10		Element is: Required **No**; Permitted **No**; Prohibited **Yes**; List all the registered GDPs that are allowed: List all profile-defined GDPs that are allowed and attach complete description: Other:
T.19.11	Same as Model Profile: **Yes**	
RECTANGLE [v1] References: 7.6.11 T.14.3 D.2.2.2		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-area geometric degeneracies shall be as defined in T.14.3. Other: None.
T.19.12	Same as Model Profile: **Yes**	
CIRCLE [v1] References: 7.6.12 T.14.3 D.2.2.2		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-area geometric degeneracies shall be as defined in T.14.3. Other: ***None.***
T.19.13	Same as Model Profile: **Yes**	
CIRCULAR ARC 3 POINT [v1] References: 7.6.13 T.14.2 D.2.2.2 D.4.5.4		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***

T.19.14	Same as Model Profile: **Yes**	
CIRCULAR ARC 3 POINT CLOSE [v1] References: 7.6.14 T.14.3 D.2.2.2 D.4.5.5		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-area geometric degeneracies shall be as defined in T.14.3. Other: ***None.***
T.19.15	Same as Model Profile: **Yes**	
CIRCULAR ARC CENTRE [v1] References: 7.6.15 T.14.2 D.2.2.2 D.4.5.6		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.16	Same as Model Profile: **Yes**	
CIRCULAR ARC CENTRE CLOSE [v1] References: 7.6.16 T.14.3 D.2.2.2 D.4.5.7		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-area geometric degeneracies shall be as defined in T.14.3. Other: ***None.***
T.19.17	Same as Model Profile: **Yes**	

ELLIPSE [v1] References: 7.6.17 T.14.3 D.2.2.2 D.4.5.9 D.4.5.10		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-area geometric degeneracies shall be as defined in T.14.3. Other: ***None.***
T.19.18	Same as Model Profile: **Yes**	
ELLIPTICAL ARC [v1] References: 7.6.18 T.14.2 D.2.2.1 D.4.5.11		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.19	Same as Model Profile: **Yes**	
ELLIPTICAL ARC CLOSE [v1] References: 7.6.19 T.14.3 D.2.2.2 D.4.5.12		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-area geometric degeneracies shall be as defined in T.14.3. Other: ***None.***
T.19.20	Same as Model Profile: **Yes**	

CIRCULAR ARC CENTRE REVERSED [v2] References: 7.6.20 T.14.2 D.2.2.1 D.4.5.8		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.21	Same as Model Profile: **Yes**	
CONNECTING EDGE [v2] References: 7.6.21 T.14.2 D.2.2.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; This element shall be permitted only if BEGIN/END FIGURE is permitted. Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.22	Same as Model Profile: **No**	
HYPERBOLIC ARC [v3] References: 7.6.22 T.14.2 D.2.2.1	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.23	Same as Model Profile: **No**	

PARABOLIC ARC [v3] References: 7.6.23 T.14.2 D.2.2.1	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.24	Same as Model Profile: **Yes**	
NON-UNIFORM B-SPLINE [v3] References: 7.6.24 T.14.2 D.2.2.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Set of spline orders: ***cubic spline (order=4).*** Maximum number of control points: ***4096.*** Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***The spline shall be clamped form, i.e., the first 4 knots shall be identical and the last 4 knots shall be identical.***
T.19.25	Same as Model Profile: **Yes**	
NON-UNIFORM RATIONAL B-SPLINE [v3] References: 7.6.25 T.14.2 D.2.2.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Set of spline orders: ***cubic spline (order=4).*** Maximum number of control points: ***4096.*** Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***The spline shall be clamped form, i.e., thefirst 4 knots shall be identical and the last 4 knotsshall be identical.***
T.19.26	Same as Model Profile: **Yes**	

POLYBEZIER [v3] References: 7.6.26 T.14.2 D.2.2.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum number of points: ***4096.*** Any restrictions on the continuity indicator? ***None.*** Zero-length geometric degeneracies shall be as defined in T.14.2. Other: ***None.***
T.19.27	Same as Model Profile: **Yes**	
POLYSYMBOL [v3] References: 7.6.27 D.2.2.1		Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Point list: Effect of a reference to a symbol index parameter which is not in the symbol library. Other: ***NOTE - This element is prohibited because SYMBOL LIBRARY LIST is prohibited.***
T.19.28	Same as Model Profile: **No**	
BITONAL TILE [v3] References: 7.6.28 D.2.2.1 D.4.5.13	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; List allowable compression types: ***2, 5, or 6.*** Requirements on row padding: ***None.*** Other: ***The WebCGM 1.0 values 0, 1 are deprecated.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; List allowable compression types: ***Values 0..6.*** Requirements on row padding: ***None.*** Other: ***CCITT compression methods (T6 and T4) should be used with 1 bit cell colour precision and indexed colour.*** **Note — Several compression types have beenregistered (as of date of publication) in the ISORegister of Graphical Items, specifically: JPEG,LZW, and PNG.**
T.19.29	Same as Model Profile: **No**	

TILE [v3] References: 7.6.29 D.2.2.1 D.4.5.13	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; List allowable compression types: ***5, 6, 7, or 9*** Requirements on row padding: ***None.*** Other: ***The value 9 is the ISO registered value for compression method 0 of PNG.*** ***The WebCGM 1.0 values 0, 1, 2 are deprecated.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; List allowable compression types: ***Values 0..6.*** Requirements on row padding? ***None.*** Other: ***CCITT compression methods (T6 and T4) should be used with 1 bit cell colour precision and indexed colour.*** **Note — Several compression types have been registered (as of date of publication) in the ISORegister of Graphical Items, specifically: JPEG, LZW, and PNG.**

6.9 Attribute Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.20.1	Same as Model Profile: **No**	
LINE BUNDLE INDEX [v1] References: 7.7.1 9.5.4.2 D.4.6.1 T.17.11	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; The *line bundle index* parameter shall follow the rules for indexes, clause 7.5.4.2. For [v1] metafiles, allowable index values: For [v2/3] metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *line bundle index* parameter shall follow the rules for indexes, clause 7.5.4.2. For [v1] metafiles, allowable index values: ***1..5.*** • **index 1 2 3 4 5** **line type 1 2 3 4 5** **line width 1.0 1.0 1.0 1.0 1.0** **line colour 1 1 1 1 1** For [v2], [v3], and [v4]metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***
T.20.2	Same as Model Profile: **No**	
LINE TYPE [v1] References: 7.7.2 9.4.17 D.4.6.2	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Select 1 or more of the following: • values 1..5: **Yes** • subset of registered values (attach list): 6..15: **Yes** • profile-defined values (attach complete description): **No** For [v3] and [v4] metafiles, • negative values assigned by the LINE AND EDGE TYPE DEFINITION element. **Yes**;	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Select 1 or more of the following: • values 1..5; **Yes** • subset of registered values (attach list); **No** • profile-defined values (attach complete description); **No** For [v3] and [v4] metafiles, • negative values assigned by the LINE AND EDGE TYPE DEFINITION element. **Yes**

	Other: ***Line types 6-15 are included in the Register of Graphical Objects. This register is available from the ISO SC24 Committee. See Section 6.16 about specific and generic line types.***	Other: ***None.***
T.20.3	Same as Model Profile: **Yes**	
LINE WIDTH [v1] References: 7.7.3 D.4.6.3		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Is value zero allowed? (yes/no) ***Yes.*** If yes, specify its meaning.***Minimum available line width.*** Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.4	Same as Model Profile: **Yes**	
LINE COLOUR [v1] References: 7.7.4 9.5.4.1 T.14.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *line colour specifier* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.5	Same as Model Profile: **No**	
MARKER BUNDLE INDEX [v1] References: 7.7.5 9.5.4.2 T.17.12 D.4.6.1	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; The *marker bundle index* parameter shall follow the rules for indexes, clause 9.5.4.2. For [v1] metafiles, allowable index values: For [v2/3] metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *marker bundle index* parameter shall follow the rules for indexes, clause 9.5.4.2. For [v1] metafiles, allowable index values: ***1..5.*** **index 1 2 3 4 5** **marker type 1 2 3 4 5** **marker width 1.0 1.0 1.0 1.0 1.0** **marker colour 1 1 1 1 1** For [v2], [v3] and [v4]metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***

T.20.6	Same as Model Profile: **Yes**	
MARKER TYPE [v1] References: 7.7.6 D.4.6.4		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Indicate one or more of the following restrictions: • values 1..5; **Yes** • subset of registered values (attach list); **No** • profile-defined values (attach complete description). **No** Other: ***None.***
T.20.7	Same as Model Profile: **Yes**	
MARKER SIZE [v1] References: 7.7.7 D.4.6.5		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Is value zero allowed? (yes/no) ***Yes.*** If yes, specify its meaning. ***Minimum available size.*** Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.8	Same as Model Profile: **Yes**	
MARKER COLOUR [v1] References: 7.7.8 9.5.4.1 T.14.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *marker colour specifier* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.9	Same as Model Profile: **No**	
TEXT BUNDLE INDEX [v1] References: 7.7.9 9.5.4.2	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; The *text bundle index* parameter shall follow the rules for indexes, clause 9.5.4.2. • For [v1] metafiles, allowable index values: For [v2/3] metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *text bundle index* parameter shall follow the rules for indexes, clause 9.5.4.2. For [v1] metafiles, allowable index values: ***1..2.*** **index 1 2** **font index 1 1**

T.17.13 D.4.6.1		**text precision stroke stroke** **character expansion factor 1.0 0.7** **character spacing 0.0 0.0** **text colour 1 1** For [v2], [v3] and [v4] metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***
T.20.10	Same as Model Profile: **Yes**	
TEXT FONT INDEX [v1] References: 7.7.10 9.5.4.2 T.16.13		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Every referenced index shall refer to an entry in the FONT LIST (see T.16.13). Other: ***None.***
T.20.11	Same as Model Profile: **No**	
TEXT PRECISION [v1] References: 7.7.11	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***Value shall be 'stroke'.*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.12	Same as Model Profile: **Yes**	
CHARACTER EXPANSION FACTOR [v1] References: 7.7.12 D.4.6.7		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Is value zero allowed? (yes/no) ***No.*** If yes, state the meaning. Any restrictions on the parameter value? ***Values shall be restricted to the range 0.1..10.0*** Other: ***None.***

T.20.13	Same as Model Profile: **Yes**	
CHARACTER SPACING [v1] References: 7.7.13 D.4.6.8		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***Values shall be restricted to the range of -1.0..5.0.*** Other: ***None.***
T.20.14	Same as Model Profile: **Yes**	
TEXT COLOUR [v1] References: 7.7.14 9.5.4.1 T.14.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *text colour specifier* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.15	Same as Model Profile: **Yes**	
CHARACTER HEIGHT [v1] References: 7.7.15 D.4.6.9		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Is zero height allowed: (yes/no) ***Yes.*** If yes, state its meaning: Minimum available height. Any restrictions on the parameter? ***None.*** Other: ***None.***
T.20.16	Same as Model Profile: **Yes**	
CHARACTER ORIENTATION [v1] References: 7.7.16 D.4.6.10		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the following distortion aspects? rotation? **None.** skewing? **None.** mirroring? **None.** aspect ratio? **None.**

		Other: ***None.***
T.20.17	Same as Model Profile: **Yes**	
TEXT PATH [v1] References: 7.7.17 D.4.6.11		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.18	Same as Model Profile: **Yes**	
TEXT ALIGNMENT [v1] References: 7.7.18 D.4.6.12		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the horizontal and vertical alignment values? ***None.*** Any restrictions on the continuous horizontal and vertical alignment values? ***None.*** Other: ***None.***
T.20.19	Same as Model Profile: **No**	
CHARACTER SET INDEX [v1] References: 7.7.19 9.5.4.2 T.16.14 T.16.22 D.4.6.13	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Every referenced index shall refer to an entry in the CHARACTER SET LIST. This includes implicit reference to the default index value. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Every referenced index shall refer to an entry in the CHARACTER SET LIST or GLYPH MAPPING. This includes implicit reference to the default index value. Other: ***None.***
T.20.20	Same as Model Profile: **No**	

ALTERNATE CHARACTER SET INDEX [v1] References: 7.7.20 9.5.4.2 T.16.14 T.16.22 D.4.6.13	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Every referenced index shall refer to an entry in the CHARACTER SET LIST. This includes implicit reference to the default index value. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Every referenced index shall refer to an entry in the CHARACTER SET LIST or GLYPH MAPPING. This includes implicit reference to the default index value. Other: ***None.***
T.20.21	Same as Model Profile: **No**	
FILL BUNDLE INDEX [v1] References: 7.7.21 9.5.4.2 T.17.14 D.4.6.1	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; The *fill bundle index* parameter shall follow the rules for indexes, clause 9.5.4.2. For [v1] metafiles, allowable index values: For [v2/3] metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *fill bundle index* parameter shall follow the rules for indexes, clause 9.5.4.2. For [v1] metafiles, allowable index values: ***1..5.*** **index 1 2 3 4 5** **interior style hatch hatch hatch hatch hatch** **fill colour 1 1 1 1 1** **hatch index 1 2 3 4 5** **pattern index 1 1 1 1 1** For [v2], [v3] and [v4] metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***
T.20.22	Same as Model Profile: **No**	
INTERIOR STYLE [v1] References: 7.7.22 D.4.6.15	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; For 'hollow' interior style, line type and width of the bounding line: ***Solid line type and default line width.*** Any restrictions on the parameter value? ***hollow, solid, pattern, hatch, empty, interpolated.*** Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; For 'hollow' interior style, line type and width of the bounding line: ***Solid line type and default line width.*** Any restrictions on the parameter value? ***None.*** Other: ***None.***

T.20.23	Same as Model Profile: **Yes**	
FILL COLOUR [v1] References: 7.7.23 9.5.4.1 T.14.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *fill colour specifier* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.24	Same as Model Profile: **Yes**	
HATCH INDEX [v1] References: 7.4.18 7.7.24 6.7.4.3 D.4.6.16	*Note. See 6.17 for further discussion of hatch interiors in WebCGM.*	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Select 1 or more of the following: • values 1..6: **Yes** • subset of registered values (attach list): **No** • profile-defined values (attach complete description): **No** For [v3] and [v4]metafiles, • negative values assigned by the HATCH STYLE DEFINITION element. **Yes** Other: ***None.***
T.20.25	Same as Model Profile: **Yes**	
PATTERN INDEX [v1] References: 7.7.25 9.5.4.2		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *pattern index* parameter shall follow the rules for indexes, clause 9.5.4.2. Any restrictions on the parameter value? ***None***. Other: ***None***.
T.20.26	Same as Model Profile: **No**	

EDGE BUNDLE INDEX [v1] References: 7.7.26 9.5.4.2 T.17.15 D.4.6.1	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; The *edge bundle index* parameter shall follow the rules for indexes, clause 9.5.4.2. For [v1] metafiles, allowable index values: For [v2/3] metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *edge bundle index* parameter shall follow the rules for indexes, clause 9.5.4.2. For [v1] metafiles, allowable index values: ***1..5.*** • ***index 1 2 3 4 5*** **edge type 1 2 3 4 5** **edge width 1.0 1.0 1.0 1.0 1.0** **edge colour 1 1 1 1 1** For [v2], [v3] and [v4]metafiles, any referenced bundle shall have an explicit representation definition. Other: ***None.***
T.20.27	Same as Model Profile: **Yes**	
EDGE TYPE [v1] References: 7.4.17 9.7.27 D.4.6.17	*Note. See 6.16 for further discussion of line and edge type definitions in WebCGM.*	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Select 1 or more of the following: • values 1..5: **Yes** • subset of registered values (attach list): **No** • profile-defined values (attach complete description): **No** For [v3] and [v4] metafiles, • negative values assigned by the LINE AND EDGE TYPE DEFINITION element. **Yes** Other: ***None.***
T.20.28	Same as Model Profile: **Yes**	
EDGE WIDTH [v1] References: 7.7.28 D.4.6.18		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Is value zero allowed? (yes/no) ***Yes.*** If yes, specify its meaning. ***Minimum available edge width.*** Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.29	Same as Model Profile: **Yes**	

EDGE COLOUR [v1] References: 7.7.29 9.5.4.1 T.14.1		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; The *edge colour specifier* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.30	Same as Model Profile: **Yes**	
EDGE VISIBILITY [v1] References: 7.7.30		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.31	Same as Model Profile: **Yes**	
FILL REFERENCE POINT [v1] References: 7.7.31		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? None. Other: None.
T.20.32	Same as Model Profile: **Yes**	
PATTERN TABLE [v1] References: 7.7.32		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Maximum size for nx: ***32.*** Allowable values for nx: ***8, 16, or 32.*** Maximum size for ny: ***32.*** Allowable values for ny: ***8, 16, or 32.*** Any restrictions on the number of pattern definitions? ***64.*** Any restrictions on allowable combinations of nx and ny? ***None.*** Any restrictions on the number of colours? ***None.*** Other: ***None.***

T.20.33	Same as Model Profile: **Yes**	
PATTERN SIZE [v1] References: 7.7.33 D.4.6.19		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Must pattern vectors be parallel to coordinate axes? (yes/no) ***Yes.*** If no, state the meaning of skewed or non-aligned patterns. Other: ***None.***
T.20.34	Same as Model Profile: **No**	
COLOUR TABLE [v1] References: 7.7.34 9.5.4.1 T.14.1	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any limits on the length of colour list? ***Monochrome: 2, Colour: 256.*** Any restrictions on the index values? ***Index values shall not exceed the maximum colour index.*** Other: ***Greyscale metafiles are considered special cases of colour metafiles.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any limits on the length of colour list? ***Monochrome: 2, Greyscale: 64, Colour: 256.*** Any restrictions on the index values? ***Index values shall not exceed the maximum colour index.*** Other: ***None.***
T.20.35	Same as Model Profile: **No**	
ASPECT SOURCE FLAGS [v1] References: 7.7.35 D.4.6.20	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Are all ASF values to be the same: for the metafile? (yes/no) within each class (line, marker, text, fill, edge) of primitive? (yes/no) Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Are all ASF values to be the same: for the metafile? (yes/no) ***No.*** within each class (line, marker, text, fill, edge) of primitive? (yes/no) ***Yes.*** Other: ***None.***
T.20.36	Same as Model Profile: **No**	
PICK IDENTIFIER [v2] References: 7.7.36	Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the parameter value? Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***

T.20.37	Same as Model Profile: **No**	
LINE CAP [v3] References: 7.7.37 9.5.7.5	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values for the line cap indicator? (choose 1 or both) **Yes**; values 1..4; **No**; subset of registered values (attach list). Any restrictions on the set of values for the dash cap indicator? (choose 1 or both) **Yes**; values 1..3; **No**; subset of registered values (attach list). Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values for the line cap indicator? (choose 1 or both) **Yes**; values 1..5; **No**; subset of registered values (attach list). Any restrictions on the set of values for the dash cap indicator? (choose 1 or both) **Yes**; values 1..3; **No**; subset of registered values (attach list). Other: ***None.***
T.20.38	Same as Model Profile: **Yes**	
LINE JOIN [v3] References: 7.7.38 9.5.7.5 T.26.7		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values? (choose 1 or both) **Yes**; values 1..4; **No**; subset of registered values (attach list). Other: ***None.***
T.20.39	Same as Model Profile: **Yes**	
LINE TYPE CONTINUATION [v3] References: 7.7.39 9.5.7.5 T.26.7		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values? ***1..4.*** Other: ***None.***
T.20.40	Same as Model Profile: **Yes**	

LINE TYPE INITIAL OFFSET [v3] References: 7.7.40		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.41	Same as Model Profile: **Yes**	
TEXT SCORE TYPE [v3] References: 7.7.41		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values? (choose 1 or both) **Yes**; Values 1..4; **No**; Subset of registered values (attach list). Other: ***None.***
T.20.42	Same as Model Profile: **Yes**	
RESTRICTED TEXT TYPE [v3] References: 7.7.42 9.5.7.5 T.26.7		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values? (choose 1 or both) **Yes**; Values 1..6; **No**; Subset of registered values (attach list). Algorithms for achieving restriction type? (attach) ***Not specified.*** Other: ***None.***
T.20.43	Same as Model Profile: **Yes**	
INTERPOLATED INTERIOR [v3] References: 7.7.43		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any limits on the number of stages? ***Maximum number of stages is 8.*** Any restrictions on the set of values? (choose 1 or both) **Yes**; Values 1..3; **No**; Subset of registered values (attach list). Other: ***None.***

T.20.44	Same as Model Profile: **No**	
EDGE CAP [v3] References: 7.7.44 9.5.7.5 T.26.7	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values for the edge cap indicator? (choose 1 or both) **Yes**; values 1..4; **No**; subset of registered values (attach list). Any restrictions on the set of values for the dash cap indicator? (choose 1 or both) **Yes**; values 1..3; **No**; subset of registered values (attach list). Other: ***None.***	Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values for the edge cap indicator? (choose 1 or both) **Yes**; values 1..5; **No**; subset of registered values (attach list). Any restrictions on the set of values for the dash cap indicator? (choose 1 or both) **Yes**; values 1..3; **No**; subset of registered values (attach list). Other: ***None.***
T.20.45	Same as Model Profile: **Yes**	
EDGE JOIN [v3] References: 7.7.45 9.5.7.5 T.26.7		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values? (choose 1 or both) **Yes**; values 1..4; **No**; subset of registered values (attach list). Other: ***None.***
T.20.46	Same as Model Profile: **Yes**	
EDGE TYPE CONTINUATION [v3] References: 7.7.46 9.5.7.5 T.26.7		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the set of values? ***1..4.*** Other: ***None.***
T.20.47	Same as Model Profile: **Yes**	

EDGE TYPE INITIAL OFFSET [v3] References: 7.7.47		Element is: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter value? ***None.*** Other: ***None.***
T.20.48	Same as Model Profile: **Yes**	
SYMBOL LIBRARY INDEX [v3] References: 7.7.48 9.5.4.2 T.16.23		Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Every referenced index shall refer to an entry in the SYMBOL LIBRARY LIST (see T.16.23). Other: ***This element is prohibited because SYMBOL LIBRARY LIST is prohibited.***
T.20.49	Same as Model Profile: **Yes**	
SYMBOL COLOUR [v3] References: 7.7.49 9.5.4.1 T.14.1 T.16.23 D.4.6.21		Element is: Required **No**; Permitted **No**; Prohibited **Yes**; The *symbol colour specifier* parameter shall follow the rules for colour, clause 9.5.4.1 and T.14.1. Any restrictions on the parameter value? Other: ***This element is prohibited because SYMBOL LIBRARY LIST is prohibited.***
T.20.50	Same as Model Profile: **Yes**	

SYMBOL SIZE [v3] References: 7.7.50 T.16.23		Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Is value zero is allowed: (yes/no) If yes, specify its meaning. Any restrictions on the parameter value? Other: ***This element is prohibited because SYMBOL LIBRARY LIST is prohibited.***
T.20.51	Same as Model Profile: **Yes**	
SYMBOL ORIENTATION [v3] References: 7.7.51 T.16.23 D.4.6		Element is: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on rotation? Any restrictions on skewing? Any restrictions on mirroring? Any restrictions on distortion of aspect ratio? Other: ***This element is prohibited because SYMBOL LIBRARY LIST is prohibited***.

6.10. Escape Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.21.1	Same as Model Profile: **No**;	
ESCAPE [v1] References: 7.8.1	Element: Required **No**; Permitted **Yes**; Prohibited **No**; List all registered ESCAPEs that are allowed: • ***ESCAPE 22, Transparent Cell Colour [v1/v2] metafiles only. (See T.18.15).*** • ***ESCAPE 45, Alpha Transparency: The SDR parameter is encoded as a real value between 0.0 and 1.0, inclusively, and applies to all subsequent graphical primitives. (See Section 2.2.2 and 2.2.3).*** List all profile-defined ESCAPEs that are allowed and attach complete description: Other: ***All ESCAPE element parameters shall be encoded as SDRs*** NOTE: Only registered ESCAPEs and profile-defined ESCAPEs shall be allowed in profiles.	Element: Required **No**; Permitted **Yes**; Prohibited **No**; List all registered ESCAPEs that are allowed: ***ESCAPE 22, Transparent Cell Colour [v1/v2] metafiles only***. List all profile-defined ESCAPEs that are allowed and attach complete description: ***None.*** Other: ***None***

6.11 External Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.22.1	Same as Model Profile: **No**	
MESSAGE [v1] References: 7.9.1	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Values of the *action required flag* parameter: • 'action' Permitted **No**; Prohibited **No**; (if permitted, specify the messages and actions taken) • 'no action' Permitted **No**; Prohibited **No**; Any restrictions on the length of the message string, other than those for type SF parameter? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Values of the *action required flag* parameter: • 'action' Permitted **No**; Prohibited **Yes**; (if permitted, specify the messages and actions taken) • 'no action' Permitted **Yes**; Prohibited **No**; Any restrictions on the length of the message string, other than those for SF parameter? ***None.*** Other: ***None.***
T.22.2	Same as Model Profile: **No**	
APPLICATION DATA [v1] References: 7.9.2	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Attach a syntactic and semantic description of all application data elements associated with this profile. Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; The use of this element shall not be restricted. Attach a syntactic and semantic description of all application data elements associated with this profile. Other: ***None.***

6.12 Segment Elements

Element	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.23.1	Same as Model Profile: **No**	
COPY SEGMENT [v2] References: 7.10.1 D.4.9.2	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Every segment identifier shall refer to a defined segment. Any limits on the segment transformation application value? Any restrictions on the nature of the transformation (e.g., permitting only isotropic transformations)? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Every segment identifier shall refer to a defined segment. Any limits on the segment transformation application value? ***None.*** Any restrictions on the nature of the transformation (e.g., permitting only isotropic transformations)? ***Non-singular.*** Other: *None.*
T.23.2	Same as Model Profile: **No**	

INHERITANCE FILTER [v2] References: 7.10.2	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Any limits on the filter selection list? Any limits on the selection setting? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any limits on the filter selection list? ***None.*** An limits on the selection setting? ***None.*** Other: ***None.***
T.23.3	Same as Model Profile: **No**	[redacted]
CLIP INHERITANCE [v2] References: 7.10.3 D.4.9.2	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Any limits on the parameter? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any limits on the parameter? ***None.*** Other: ***None.***
T.23.4	Same as Model Profile: **No**	[redacted]
SEGMENT TRANS-FORMATION [v2] References: 7.10.4	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the nature of the transformation (e.g., permitting only isotropic transformations)? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the nature of the transformation (e.g., permitting only isotropic transformations)? ***Non-singular.*** Other: ***None.***
T.23.5	Same as Model Profile: **No**	[redacted]
SEGMENT HIGHLIGHTING [v2] References: 7.10.5	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the parameter values? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***None.*** Other: ***None.***
T.23.6	Same as Model Profile: **No**	[redacted]

SEGMENT DISPLAY PRIORITY [v2] References: 7.10.6	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the parameter values? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***None.*** Other: ***None.***
T.23.7	Same as Model Profile: **No**	
SEGMENT PICK PRIORITY [v2] References: 7.10.7	Element: Required **No**; Permitted **No**; Prohibited **Yes**; Any restrictions on the parameter values? Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Any restrictions on the parameter values? ***None.*** Other: ***None.***

6.13 Application Structure Descriptor Elements

Functionality	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.24.1	Same as Model Profile: **No**	
APPLICATION STRUCTURE ATTRIBUTE [v4] References: 6.9 6.13.5 7.9.2 7.1.1	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Define the set of structure elements for use within application structures (APS), and attach complete syntactic and semantic description: ***The set of attributes allowed is listed in Section 3.2.2, which includes complete syntactic and semantic definitions, as well as permissibility according to APS type.*** Other: ***None.***	Element: Required **No**; Permitted **Yes**; Prohibited **No**; Define the set of structure elements for use within application structures, and attach complete syntactic and semantic description: ***None.*** Other: ***None.***

6.14 Generator Implementation Requirements

Functionality	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.25.1	Same as Model Profile: **Yes**	

Colour requirements References: 9.5.4.1 9.5.6.2.2		Colour mapping is: Permitted **Yes**; Prohibited **No**; Reduction of the number of colours? ***Not specified.*** ***NOTE - If mapping of application colours to metafile colour specifications is required, it is recommended that colour distance in the mapping be computed by the Euclidean metric in CIEXYZ space.*** Definition of mapping algorithms, metrics, and colour space? ***No specific colour mapping techniques or selection of metafile colour sets are defined.*** For [v1/2] metafiles, implicit colour calibration specifications? ***No specifications are defined.*** Other: ***None.***
T.25.2	Same as Model Profile: **Yes**	
Geometric accuracy and latitude References: 9.5.6.2.1		Accuracy and latitude for mapping application graphics to CGM graphical primitive elements: Accuracy and latitude for mapping application graphics to CGM graphical primitive elements: ***Generators shall produce a metafile whose graphical primitive elements match the application graphical primitives accurately to within ±0.1% of relative position within the VDC Extent box or ±1/2 pixel of the intended size, whichever is greater. Generators shall produce geometric size aspects of the primitives (e. g., text size, line width, and edge width) to within 1% of the intended size or ±1/2 pixel of the intended size, whichever is greater.*** ***This requirement shall apply to all graphical primitive elements, unless superseded by specific element requirements in this clause.***
T.25.3	Same as Model Profile: **Yes**	
Text accuracy and latitude References: 9.5.6.2.3		Is text accuracy and latitude addressed? (yes/no) ***Yes***. If yes, specify. ***Metafile text specifications shall match the text of the application picture to within ±1% of relative to the intended size or ±1/2 pixel of the intended size, whichever is greater, for the placement and overall extent of each text string.***
T.25.4	Same as Model Profile: **No**	

Font substitution References: 9.5.6.2.4 I.2	Font substitution is: Permitted **Yes**; Prohibited **No**; Similarity of font visual characteristics? ***Substituted fonts shall be metrically equivalent or be controlled by the RESTRICTED TEXT element.*** Font metrics? ***Specified in ISO/IEC 8632:1999 Annex I.2 for the core 13 fonts.*** Individual glyph metrics? ***Specified in ISO/IEC 8632:1999 Annex I.2 for the core 13 fonts.*** Other: ***None.***	Font substitution is: Permitted **Yes**; Prohibited **No**; Similarity of font visual characteristics? ***Substituted fonts shall have similar visual characteristics (e.g., posture, weight, proportionate width).*** Font metrics? ***Specified in clause I.2.*** Individual glyph metrics? ***Specified in clause I.2.*** Other: ***None.***
T.25.5	Same as Model Profile: **Yes**	
Preservation of primitives References: 9.5.6.3		Is preservation of graphical primitive elements addressed? (yes/no) ***No.*** If yes, specify allowable substitutions.
T.25.6	Same as Model Profile: **No**	
Semantic latitude References: 9.5.6.4	Drawing priority and mode: ***Priority shall correspond to the metafile order (i.e., primitives occurring later in the file shall overlay primitives occurring earliest in the file). Mode shall be "replacement" mode.*** Clipping: ***Clipping shall be to the intersection of the clip rectangle, the VDC EXTENT, the device viewport, and the device view surface limits.*** Edge centreing: ***Edges shall be centred on the ideal mathematically-defined edge of the area*** Meaning of predefined line types and edge types: ***See Section 6.16 about specific and generic line types.*** Meaning of predefined hatch styles: ***See Section 6.17 about specific and generic hatch styles.*** Other: ***None.***	Drawing priority and mode: ***Priority shall correspond to the metafile order (i.e., primitives occurring later in the file shall overlay primitives occurring earliest in the file). Mode shall be "replacement" mode.*** Clipping: ***Clipping shall be to the intersection of the clip rectangle, the VDC EXTENT, the device viewport, and the device view surface limits.*** Edge centreing: ***Edges shall be centred on the ideal mathematically-defined edge of the area.*** Meaning of predefined line types and edge types: ***The exact on-off definitions for the predefined line types and edge types are not specified.*** Meaning of predefined hatch styles:***The inter-line spacing is not specified. Use the latitudes of annex D.4.6.16 for the angular directions.*** Other: ***None.***
T.25.7	Same as Model Profile: **Yes**	

Error processing References: 9.5.6.5		Is error processing addressed? (yes/no) ***No.*** If yes, specify the action taken. Classification of error severity? Requirements for error recovery? Requirements for error reporting? Additional areas? Other: ***None.***
T.25.8	Same as Model Profile: **Yes**	
Reporting References: 9.5.6.6		Is reporting required? (yes/no) ***No.*** If yes, specify the action taken. Method and format of the reporting? Requirement to report substitution, error, fallback behavior, mappings, or other behaviors? Additional areas? Other: ***None.***
T.25.9	Same as Model Profile: **Yes**	
Degeneracies References: 9.5.6.7 9.5.4.4 D.2 D.4		Is the generation of degenerate primitives addressed? (yes/no) ***No. The generation of degenerate primitives is not restricted.*** If yes, attach specifications. Other: ***None.***

6.15 Interpreter Implementation Requirements

Functionality	Specifications - WebCGM 2.1 Profile	Specifications - Model Profile
T.26.1	Same as Model Profile: **Yes**	

Number of pictures References: 9.5.7.2 T.13.2		If 0 pictures are permitted (see T.13.2), describe the interpreter behavior: ***Prohibited by T.13.2.***
T.26.2	Same as Model Profile: **Yes**	
Empty pictures References: 9.5.7.3 T.13.3		If permitted (see T.13.3), interpreter behavior: ***The graphical effect shall be one picture in the background colour.***
T.26.3	Same as Model Profile: **Yes**	
Colour requirements References: 9.5.4.1 9.5.7.4.2 9.5.4.5		Interpreters shall be classified as either monochrome, greyscale, or colour interpreters (depending on the colour capability of the interpreter), and shall meet the criteria in attachment 26.3 ***Conversions between different colour models shall be according to the conversions in annex G.*** Mapping of metafile colour to device components? ***If mapping (to fewer colour, or greyscale, or monochrome) is required for RGB metafiles, the recommendations of annex D.3.2 shall be used.*** For [v1/2] metafiles, implicit colour calibration specifications? ***No specifications are defined.*** Other: ***None.***
T.26.4	Same as Model Profile: **Yes**	
Geometric accuracy and latitude References: 9.5.7.4.1		Accuracy and latitude for placement and realization of geometric aspects when geometric primitive elements are rendered. ***Interpreters shall render graphical primitive elements accurately to within ± 0.1% of relative position within the VDC Extent box or ±1/2 of the pixel resolution of the output device, whichever is greater. Interpreters shall render the geometric size aspect of primitives (e.g., text size, line width, and edge width) to within 1% of the intended size or ±1/2 pixel of resolution of the output device, whichever is greater.*** This requirement shall apply to all graphical primitive elements, unless superseded by specific element requirements in this clause.

T.26.5	Same as Model Profile: **Yes**	
Text rendering References: 9.5.7.4.3		Is text accuracy and latitude addressed? (yes/no) ***Yes.*** If yes, specify. ***Interpreter-rendered text shall match the text specification of the metafile to within 1% relative to the intended size or ±1/2 pixel of resolution of the output device, whichever is greater, for the placement and overall extent of each text string.*** Is precision of text rendering is addressed? (yes/no) ***Yes.*** If yes, specify interpreter action. ***Interpreters shall render text using 'stroke' precision, regardless of the actual value of the TEXT PRECISION of the metafile.***
T.26.6	Same as Model Profile: **No**	
Font substitution References: T.16.13 9.5.7.4.4 annex I.2	Font substitution is: Permitted **Yes**; Prohibited **No**; If prohibited, use the font as specified in the FONT LIST. If permitted, include a reference set of font and glyph metrics which correspond to the canonical instances of the substitutable font. ***See the FONT LIST element and annex I.2 CGM:1999.*** Are substitution methods, latitudes, and constraints addressed? (yes/no) **No** If yes, specify: Similarity of font visual characteristics? ***Substituted fonts shall be metrically equivalent or be controlled by the RESTRICTED TEXT element.*** Font metrics? ***Substituted fonts shall have similar metrics to the fonts specified in the metafile.*** Individual glyph metrics? ***Specified in ISO/IEC 8632:1999 Annex I.2 for the core thirteen fonts.*** Additional areas? ***None***. Other: ***A method for the user specification of font substitution is described in the Application Configurable Item chapter.***	Font substitution is: Permitted **Yes**; Prohibited **No**; If prohibited, use the font as specified in the FONT LIST. If permitted, include a reference set of font and glyph metrics which correspond to the canonical instances of the substitutable font. ***See the FONT LIST element and annex I.2***. Are substitution methods, latitudes, and constraints addressed? (yes/no) ***Yes.*** If yes, specify: Similarity of font visual characteristics? ***Substituted fonts shall have similar visual characteristics to the fonts specified in the metafile*** Font metrics? ***Substituted fonts shall have similar metrics to the fonts specified in the metafile.*** Individual glyph metrics? ***As specified in annex I.2.*** Additional areas? ***None***. Other: ***None.***
T.26.7	Same as Model Profile: **No**	

Semantic latitude References: 9.5.7.5 T.20.37 T.20.38 T.20.39 T.20.42 T.20.44 T.20.45 T.20.46 T.18.15	Drawing priority and mode: ***Priority shall correspond to the metafile order (i.e., primitives occurring later in the file shall overlay primitives occurring earliest in the file. Mode shall be "replacement" mode.)*** View surface clearing at picture start: ***Surface will be cleared upon the occurrence of BEGIN PICTURE BODY, except as specified elsewhere in this profile.*** Clipping: ***When CLIP INDICATOR is 'off', clipping shall be to the intersection of the device viewport and the device view surface limits. When CLIP INDICATOR is 'on', clipping shall be to the intersection of the clip rectangle, the VDC EXTENT, the device viewport, and the device view surface limits*** Edge centreing: ***Edges shall be centred on the ideal mathematically-defined edge of the area.*** Meaning of predefined line types and edge types: ***See Section 6.16 about specific and generic line types.*** Meaning of predefined hatch styles: ***See Section 6.17 about specific and generic hatch styles.*** In the absence of a LINE/MARKER/TEXT/EDGE CLIPPING MODE element, the interpreter treatment of LINE/MARKER/ TEXT/EDGE CLIPPING MODE shall be: In the style of one specific parameter value, from the set of standardized values. **YES.** Specify which one: **SHAPE** In the style of any of the specific parameter values, from the set of standardized values. **NO** For [v1/v2] metafiles, text restriction method for RESTRICTED TEXT elements, chosen from the set of standard and registered styles of the RESTRICTED TEXT TYPE element: ***Value 2.*** For [v1/2] metafiles, interpreter treatment of the 2 aspects of line cap shall be either: • in the style of one specific parameter value pair from the set of standard and registered values (excluding values 1) of the LINE CAP element. **No** Values = ? • in the style of any parameter value pair from the set of standard and registered values (excluding values 1) of the LINE CAP element. **Yes** For [v1/2] metafiles, interpreter treatment of the 2 aspects of edge cap shall be either: • in the style of one specific parameter value pair, from the set of standard and registered values (excluding values 1) of the EDGE CAP element. **No** Values = ? • in the style of any parameter value pair, from the set of standard and registered values (excluding values 1) of the EDGE CAP element. **Yes** For [v1/2] metafiles, interpreter treatment of line join shall be either: • in the style of one specific parameter value, from the	Drawing priority and mode: ***Priority shall correspond to the metafile order (i.e., primitives occurring later in the file shall overlay primitives occurring earliest in the file. Mode shall be "replacement" mode.)*** View surface clearing at picture start: ***Surface will be cleared upon the occurrence of BEGIN PICTURE BODY.*** Clipping: ***When CLIP INDICATOR is 'off', clipping shall be to the intersection of the device viewport and the device view surface limits. When CLIP INDICATOR is 'on', clipping shall be to the intersection of the clip rectangle, the VDC EXTENT, the device viewport, and the device view surface limits.*** Edge centering: ***Edges shall be centred on the ideal mathematically-defined edge of the area.*** Meaning of predefined line types and edge types: ***The exact on-off definitions for the predefined line types and edge types are not specified.*** Meaning of predefined hatch styles: ***The inter-line spacing is not specified. Use the latitudes of annex D.4.6.16 for the angular directions.*** In the absence of a LINE/MARKER/TEXT/EDGE CLIPPING MODE element, the interpreter treatment of LINE/MARKER/ TEXT/EDGE CLIPPING MODE shall be (check one): In the style of one specific parameter value, from the set of standardized values. **No.** Specify which one: In the style of any of the specific parameter values, from the set of standardized values. **Yes.** Specify which one: For [v1/v2] metafiles, text restriction method for RESTRICTED TEXT elements, chosen from the set of standard and registered styles of the RESTRICTED TEXT TYPE element: ***Value 2.*** For [v1/2] metafiles, interpreter treatment of the 2 aspects of line cap shall be either: • **No**; in the style of one specific parameter value pair from the set of standard and registered values (excluding values 1) of the LINE CAP element. Values = ? • **Yes**; in the style of any parameter value pair from the set of standard and registered values (excluding values 1) of the LINE CAP element. For [v1/2] metafiles, interpreter treatment of the 2 aspects of edge cap shall be either: • **No**; in the style of one specific parameter value pair, from the set of standard and registered values (excluding values 1) of the EDGE CAP element. Values = ? • **Yes**; in the style of any parameter value pair, from the set of standard and registered values (excluding values 1) of the EDGE CAP element. For [v1/2] metafiles, interpreter treatment of line join shall be either:

	set of standard and registered values (excluding value 1) of the LINE JOIN element. **No** Value = ? • in the style of any parameter value, from the set of standard and registered values (excluding value 1) of the LINE JOIN element. **Yes** For [v1/2] metafiles, interpreter treatment of edge join shall be either: • in the style of one specific parameter value, from the set of standard and registered values (excluding value 1) of the EDGE JOIN element. **No** Value = ? • in the style of any parameter value, from the set of standard and registered values (excluding value 1) of the EDGE JOIN element. **Yes** For [v1/2] metafiles, interpreter treatment of line type continuation shall be either: • in the style of one specific parameter value, from the set of standard and registered values (excluding value 1) of the LINE TYPE CONTINUATION element. **No** Value = ? • in the style of any parameter value, from the set of standard and registered values (excluding value 1) of the LINE TYPE CONTINUATION element. **Yes** For [v1/2] metafiles, interpreter treatment of edge type continuation shall be either: • in the style of one specific parameter value, from the set of standard and registered values (excluding value 1) of the EDGE TYPE CONTINUATION element. **No** Value = ? • in the style of any parameter value, from the set of standard and registered values (excluding value 1) of the EDGE TYPE CONTINUATION element. **Yes** Other: ***Mitre Limit handling:*** The handling of MITRE LIMIT in CGM:1999 6.5.6 is considered to contain errors, and an ISO erratum is being pursued. The following variation shall be considered conforming for the WebCGM profile, and is the preferred method when mitred line joins are rendered. 1. When the projected join point would exceed the mitre length, measured from the intersection of the inside edges of the lines at the join, then the join is rendered as a bevel style. (CGM:1999 says that the projecting point is truncated at the mitre length). 2. Any value of MITRE LIMIT that is less than 1.0 shall be mapped to 1.0. ***Note: The semantic latitude for many of these cases is addressed in the Application Configurable Items chapter, allowing explicit defintion of rendering behavior for V1/V2 files and for those V3 and later files that have not explicitly specified parameter values.***	• **No**; in the style of one specific parameter value, from the set of standard and registered values (excluding value 1) of the LINE JOIN element. Value = ? • **Yes**; in the style of any parameter value, from the set of standard and registered values (excluding value 1) of the LINE JOIN element. For [v1/2] metafiles, interpreter treatment of edge join shall be either: • **No**; in the style of one specific parameter value, from the set of standard and registered values (excluding value 1) of the EDGE JOIN element. Value = ? • **Yes**; in the style of any parameter value, from the set of standard and registered values (excluding value 1) of the EDGE JOIN element. For [v1/2] metafiles, interpreter treatment of line type continuation shall be either: • **No**; in the style of one specific parameter value, from the set of standard and registered values (excluding value 1) of the LINE TYPE CONTINUATION element. Value = ? • **Yes**; in the style of any parameter value, from the set of standard and registered values (excluding value 1) of the LINE TYPE CONTINUATION element. For [v1/2] metafiles, interpreter treatment of edge type continuation shall be either: • **No**; in the style of one specific parameter value, from the set of standard and registered values (excluding value 1) of the EDGE TYPE CONTINUATION element. Value = ? • **Yes**; in the style of any parameter value, from the set of standard and registered values (excluding value 1) of the EDGE TYPE CONTINUATION element. Other: ***None.***
T.26.8	Same as Model Profile: **Yes**	

Error processing References: 9.5.7.6		Is error processing addressed? (yes/no) ***No.*** If yes, specify the action taken. Classification of error severity? Requirements for error recovery? Requirements for error reporting? Additional areas? Other: ***None.***
T.26.9	Same as Model Profile: **Yes**	
Reporting References: 9.5.7.7		Is reporting required? (yes/no) ***No.*** If yes, specify the action taken. Method and format of the reporting? Requirement to report any substitution, error, fallback behavior, mappings, or other behaviors? Additional areas? Other: ***None.***
T.26.10	Same as Model Profile: **Yes**	
Degeneracies References: 9.5.7.8 9.5.4.4 D.2 D.4	***Note: For degenerate ELLIPTICAL ARC CLOSE, the intent of D.4.5.12 is that the radius is drawn along the coincident start-end rays.***	Is the interpretation of degenerate primitives addressed? (yes/no) ***Yes.*** If yes, for each primitive, specify the degeneracy including its source (i.e., intrinsic or computational). ***Intrinsically degenerate primitives shall be rendered as specified in annex D subsections: D.2.2, D.2.3, D.4.5.4 through D.4.5.8, D.4.5.11, and D.4.5.12. Interpreters are not required to detect computational degeneracy. If interpreters do detect computational degeneracies, they shall be rendered as specified in annex D subsections: D.2.2, D.2.3, D.4.5.4 through D.4.5.8, D.4.5.11, and D.4.5.12*** Other: ***None.***
T.26.11	Same as Model Profile: **Yes**	

Transparency References: 7.5.3 7.5.4 T.18.4		If Transparency permitted specify interpreter behavior: ***Interpreters shall implement the AUXILIARY COLOUR and TRANSPARENCY elements as described in the 2nd and 3rd paragraphs of the description in 7.5.4.*** Other: ***None.***
T.26.12	Same as Model Profile: **No**	
INTERPRETATION OF STRUCTURES AND DIRECTORIES [v4] References:	Any requirements on the interpretation of the application structures? ***Interpreters shall produce the correct graphical results.*** Is application meaning associated with application structures? yes/no ***Yes***. If yes, specify the interpreter action or actions for each type of structure. ***Viewer behavior for application structures is specified in WebCGM sections 3.2.1.1, 3.2.1.2, 3.2.1.3, 3.2.1.4, and 3.2.1.5.*** Other: None.	Any requirements on the interpretation of the application structures? ***Interpreters shall produce the correct graphical results.*** Is application meaning associated with application structures? yes/no ***No***. If yes, specify the interpreter action or actions for each type of structure. Other: ***None.***

Attachment 26.3

Colour requirements, Model Profile:

The colour mapping step (CMS) and colour rendering step (CRS) for each class of interpreters is as follows:

- monochrome:
  CMS
  all foreground information is mapped to one colour, background information to another colour.
  CRS
  all foreground information is mapped to one colour, background information to another colour.
- greyscale:
  CMS
  32 gray levels, the recommendations of annex D.3.2 is used to map colour to gray.
  CRS
  a unique representation of each of the levels of gray.
- full colour:
  CMS
  5R,9G,5B grid of RGB colour cube, plus a 32 gray levels (0-1), some of which are already on the grid.
  CRS
  a unique representation of the 254 (255) "colours".

6.16 Line and Edge Style Definitions

WebCGM supports both generic, but imprecise line types, and specific, precise line types. The realizations of line types 1..5 are described in general terms in the CGM standard (e.g., "dash-dot-dot"), and the realizations and constraints of the registered line types 6..15 are described in the ISO Register of Graphical Items (e.g., requirements for inking vertexes in

certain engineering line types). Otherwise, the exact line patterns of implicit line types 1..15 are unconstrained. Where exact realizations of line types are expected and required, the LINE AND EDGE TYPE DEFINITION element should be used.

6.17 Hatch Style Definitions

WebCGM supports both generic, but imprecise hatch styles, and specific, precise hatch styles. The realizations of hatch styles 1..6 are described in general terms in the CGM standard. Otherwise, the exact hatch patterns of implicit hatch styles 1..6 are unconstrained. Where exact realizations of hatch styles are expected and required, the HATCH STYLE DEFINITION element should be used.

6.18 JPEG Compression within the Tile Element

This section is informative (non-normative).

Following is an informative summary of the normative requirements for JPEG in WebCGM, as defined in the JPEG item in the *ISO International Register of Graphical Items*.

This profile allows the use of JPEG restricted to the TILE element. The method is limited to BASELINE JPEG. BASELINE JPEG conforms to the process required for all DCT-based decoders. The colour selection mode of the TILE element shall always be direct, independent of the COLOUR SELECTION MODE in effect in the CGM. The cell colour precision parameter of the TILE shall always be 8-bit for BASELINE JPEG. The COLOUR model of the TILE element shall be defined in the method specific parameters element of the TILE. It can be the same or independent of the COLOUR MODEL of the CGM. BASELINE JPEG shall assume that the order of the spectral bands is the same order given by the colour model as defined by the method specific parameters. For example, if the model is RGB, each scan will compress the red component, followed by the green component, followed by the blue component. For the case where the colour model is "RGB related", the specific colour model shall be defined in the method specific parameters of the TILE element. The method specific parameters shall be present for each image compressed using BASELINE JPEG. The parameters shall be encoded as an SDR. The JPEG colour model parameter is required and is specified according to the rules of the INDEX PRECISION element. Valid values are:

- 0 - JPEG colour model is the same as the COLOUR MODEL of the CGM.
- 1 - RGB
- 2 - CIELAB
- 3 - CIELUV
- 4 - CMYK
- 5 - RGB related

Values outside the range of 0-5 are not allowed. The JPEG colour submodel is required only when the JPEG colour model is "RGB related" and is specified according to the rules of the INDEX PRECISION ELEMENT. Valid values are:

- 0 - YCbCr
- 1 - YCrCb
- 2 - YUV
- 3 - YIQ
- 4 - YES
- 5 - ADT

Other values are not allowed.

Back to top of chapter

WebCGM 2.1 — Conformance

Contents

7. Conformance

This section and its subsections are normative, unless otherwise indicated.

7.1 Conformance definitions

WebCGM 2.1 defines conformance for these classes of product:

- WebCGM 2.1 instances
- WebCGM viewers (static)
- WebCGM viewers (dynamic), including WebCGM DOM implementation
- XML Companion File (XCF) instances
- Application Configurable Items (ACI) file instances

WebCGM contains both static graphics functionality and dynamic-behaviors functionality. Viewer conformance to the static graphics functionality can be measured for any kind of WebCGM viewer. Full viewer conformance to the dynamic behaviors specifications can only be measured in an environment of HTML-based documents and Web browsers. Therefore, full dynamic conformance of a viewer to all specifications in WebCGM 2.1 can only be measured for a WebCGM browser plugin (or equivalent architecture).

7.2 Deprecated and obsolete features

7.2.1 Obsolete features

7.2.1.1 Obsolete in 2.0

The following WebCGM 1.0 features, deprecated in an earlier release of WebCGM, were made obsolete in WebCGM 2.0, and are not part of the WebCGM 2.0 standard nor of this WebCGM 2.1 standard:

- multiple pictures -- a valid WebCGM 2.0 or WebCGM 2.1 instance may only contain one CGM picture; WebCGM 1.0 allowed multiple pictures.
- symbol libraries -- this 1.0 functionality was unused and unseen in the five years between WebCGM 1.0 and 2.0, therefore all elements associated with Symbol Libraries were removed from WebCGM 2.0 and not part of WebCGM 2.1.
- continued Application Structures -- disallowed in WebCGM 2.0, and not allowed in WebCGM 2.1

7.2.1.2 Obsolete in 2.1

The following WebCGM 2.0 features, deprecated in an earlier release of WebCGM, were made obsolete in WebCGM 2.1, and are not part of WebCGM 2.1:

- TILE compression types 0, 1, 2
- BITONAL TILE compression types 0, 1
- the 'viewport' param element (unused and unseen since publication of WebCGM 1.0.)
- the three WebCGM 1.0 object behaviors in the IRI fragment -- highlight, view_context, highlight_all. (These were deprecated and replaced in 2.0 by a set of behaviors that are atomic, orthogonal and comprehensive in combination.)

Note: viewers that only claim WebCGM 2.1 compliance need not handle 2.1-obsoleted features; however, viewers that claim full backward-compatible support of earlier WebCGM versions do have support requirements.

In the case of the three object behaviors, in WebCGM 2.0 the following requirement supplemented the general defined requirements for deprecated features: WebCGM 2.0 viewers were required to support these behaviors. Such support shall be according to the defined mapping onto the 2.1 set of object behaviors. Note. This specification is made because legacy occurrences of these behaviors can originate

in non-CGM content types, and can occur independently of the versioning mechanism of WebCGM content.

7.2.2 Deprecated features

The following WebCGM features are deprecated in WebCGM 2.1, and may be removed (made obsolete) in some future version:

- the 'background' param element (unused and unseen since publication of WebCGM 1.0.)
- the WebCGM 1.0 Character Set List designation tails for UTF-8 and UTF-16 were replaced by correct derivations in WebCGM 2.0, and the 1.0 forms are deprecated.

For WebCGM 2.1, the following general **definition of deprecation** applies:

- For WebCGM content: Deprecated features must not be present in conforming 2.1 content, but must be supported by conforming 2.1 viewers that support conforming 2.0 content.
- For other content types: Deprecated features should not be used in new non-CGM content (e.g., HTML content); but must must be supported by conforming 2.1 viewers that support conforming 2.0 usage.

7.3 Optional features

There are no optional features in WebCGM. Conforming static implementations must implement all static functionality as defined herein. Conforming dynamic implementations must implement all dynamic functionality, including DOM and XCF functionality, as defined herein.

7.4 Extensibility

7.4.1 Extensibility by implementations

For WebCGM implementations, the following extensibility rules apply to the given WebCGM components for which WebCGM defines conformance.

Metafiles

Metafiles are absolutely not extensible. There shall be no content in conforming WebCGM metafile instances beyond what is defined and allowed by the WebCGM Proforma of Chapter 6.

DOM

A conforming WebCGM DOM implementation must implement the interfaces of WebCGM DOM definition (Chapter 5) exactly as described therein. Any DOM implementation, whether profile defined or private (vendor defined), that extends, subsets, or modifies the WebCGM DOM is not a conformant WebCGM DOM implementation. The specification of a DOM based on or derived from

the WebCGM DOM is considered to be a new, independent DOM that would be outside of the scope of the WebCGM specification. Such a DOM would not be WebCGM conformant, and a WebCGM DOM implementation is not expected to handle this DOM.

Companion files (XCF)

XML Companion Files (XCF) are extensible with application-specific metadata in foreign namespaces, following the extension rules defined in Chapter 5, *XML Companion File*. Such namespace extensions shall have no graphical effects, i.e., if the namespace extensions are stripped from a companion file, then the graphical rendering following the load and apply of that XCF shall be the match the rendering following the load-and-apply of the unaltered XCF.

Configuration files (ACI)

The ACI file is not extensible. There shall be no content in conforming ACI file instances beyond what is defined and allowed by the Appliction Configuration Items chapter.

7.4.2 Extensibility by profiles

This sub-section is informative (non-normative.)

One design goal of WebCGM is to serve as a foundation profile for a family of closely related technical application sectors. The aim is that those sectors may succinctly present their profile definitions as delta documents from WebCGM, as explained in *Cascading Profiles*. The following rules should be observed by such profiles.

Metafiles

Profiles typically define their valid metafile content to be a subset of the full WebCGM Proforma (Chapter 6). Other than subsetting values and elements, profiles should not modify any standard WebCGM content. Profiles may extend standard WebCGM content by using the defined CGM:1999 extension mechanisms (ESCAPE, GDP, APPLICATION DATA), provide the constraints of CGM:1999 Rules for Profiles (clause 9) are observed. Specifically, such extensions should be either *profile defined* (sufficient for universal unambiguous implementation) or *registered* (in the ISO Registry of Graphical Items). Note: Profiles should use caution when extending valid metafile content, as it fragments implementations and creates interoperability problems with other application sectors.

DOM

Profiles based on WebCGM should not modify or add to the standardized WebCGM DOM methods or interfaces. Neither should such profiles add entirely new interfaces. A profile-defined DOM based on or derived from the WebCGM DOM is considered to be a new, independent DOM. A WebCGM DOM implementation should not be expected to interoperate with this DOM. The recommendations of this paragraph should help to minimize the interoperability differences amongst closely-related technical constituencies whose profiles derive from WebCGM.

Companion files (XCF)

Profiles may subset WebCGM WebCGM's XML Companion File (XCF) definition. Profiles may extend WebCGM's XCF definition with application-specific metadata in foreign namespaces, following the extension rules defined in Chapter 5, *XML Companion File*. As for implementation-defined XCF extensions, profile-defined XCF extensions shall be non-graphical.

Configuration files (ACI)

The ACI file is not extensible. There shall be no content in conforming ACI file instances beyond what is defined and allowed by the Appliction Configuration Items chapter.

7.5 Normativitity

7.5.1 Normative and informative content

The sections and subsections of this specification are labeled, after the section heading, to specify whether they are normative or informative. If a subsection is not labeled, it has the same normativity as its parent section.

For example, this conformance clause (Appendix A) says, right after the section heading, "*This section and its subsections are normative, unless otherwise indicated*." Section 7.4.1 has no label, so it is normative, while 7.4.2 says "*This sub-section is informative (non-normative.)*"

All **examples** in this specification are informative. All **illustrations** in this specification are informative. All **EBNF** in this specification is normative, unless specifically labeled as informative. All **DTDs and DTD fragments** are normative, unless specifically labeled as informative.

7.5.2 Normative language and conformance requirements

The individual conformance requirements of this specification are presented in these principal ways:

- RFC 2119 Keywords: See section 1.1 about interpretation of the RFC keywords when they occur in normative sections of this specification. Occurrences of these words in normal lowercase have the RFC-2119-defined normative implications.
- Descriptive language: Descriptive prose that defines a graphical or dynamic effect to be achieved, when occurring in a normative section, represents conformance requirements. For example: "A non-visible object is not displayed."
- Imperative voice: Imperative voice statements that define a graphical or dynamic effect to be achieved, when occurring in a normative section, represents conformance requirements. For example: "If the picture behavior value is any valid name string other than the above reserved names, (it begins with an alphabetic character (a-zA-Z)), remove the existing content (picture or document) from the frame whose name matches the string and display the specified content in the specified frame."

7.6 Validation tools

This subsection is informative (non-normative).

One of the benefits of using any CGM profile is the ability to insure interoperability through the use of

validation tools against CGM instances and certification services for applications. Once an application has been certified through a testing service, behavior of that application is predictable under the constraints of the profile. Validation and certification tools and services which exist (or have existed) and can be leveraged for WebCGM are:

- Validation tools exist for metafile *instances*.
- A WebCGM 1.0 test suite.
- A WebCGM 2.0 test suite, currently being upgraded to 2.1.
- Previous test suites and certification services for *interpreters* for the closely related ATA profile -- several viewer, printer, and interpreter products that now also support WebCGM were certified for ATA compliance.

Back to top of chapter

---

8. ECMAScript binding

This chapter is normative.

```
Prototype Object WebCGMException
    The WebCGMException class has the following constants:
        WebCGMException.INDEX_SIZE_ERR
            This constant is of type Number and its value is 1.
        WebCGMException.WEBCGMSTRING_SIZE_ERR
            This constant is of type Number and its value is 2.
        WebCGMException.INVALID_CHARACTER_ERR
            This constant is of type Number and its value is 3.
        WebCGMException.NO_DATA_ALLOWED_ERR
            This constant is of type Number and its value is 4.
        WebCGMException.NO_MODIFICATION_ALLOWED_ERR
            This constant is of type Number and its value is 5.
        WebCGMException.NOT_SUPPORTED_ERR
            This constant is of type Number and its value is 6.
        WebCGMException.INVALID_ACCESS_ERR
            This constant is of type Number and its value is 7.
        WebCGMException.FILE_NOT_FOUND_ERR
            This constant is of type Number and its value is 8.
        WebCGMException.FILE_INVALID_ERR
            This constant is of type Number and its value is 9.

Object WebCGMException
    The WebCGMException object has the following properties:
        code
            This property is of type Number.

Object WebCGMRect
    The WebCGMRect object has the following properties:
        xll
            This property is of type Number.
        yll
            This property is of type Number.
        xur
            This property is of type Number.
        yur
            This property is of type Number.

    The WebCGMRect object has the following methods:
        union(r)
            This method returns a WebCGMRect.
            The r parameter is of type WebCGMRect.

Object WebCGMMatrix
```

```
    The WebCGMMatrix object has the following properties:
        a
            This property is of type Number.
        b
            This property is of type Number.
        c
            This property is of type Number.
        d
            This property is of type Number.
        e
            This property is of type Number.
        f
            This property is of type Number.

The WebCGMMatrix object has the following methods:
        multiply(m)
            This method returns a WebCGMMatrix.
            The m parameter is of type WebCGMMatrix.
        inverse()
            This method returns a WebCGMMatrix.  // Raises WebCGMException if matrix is not
invertible.
        translate(x,y)
            This method returns a WebCGMMatrix.
            The x parameter is of type Number.
            The y parameter is of type Number.
        scale(sx,sy)
            This method returns a WebCGMMatrix.
            The sx parameter is of type Number.
            The sy parameter is of type Number.
        rotate(angle,rx,ry)
            This method returns a WebCGMMatrix.
            The angle parameter is of type Number.
            The rx parameter is of type Number.
            The ry parameter is of type Number.

Object GetWebCGMDocument
    The GetWebCGMDocument object has the following methods:
        getWebCGMDocument()
            This method returns a WebCGMMetafile.
    getAppName()
            This method returns a String.
    getAppVersion()
            This method returns a String.

Object WebCGMMetafile
    The WebCGMMetafile object has the following properties:
        metafileDescription
            This read-only property is of type String.
        firstPicture
            This read-only property is a WebCGMPicture object.
        metafileID
            This read-only property is of type String.
        metafileVersion
            This read-only property is of type Number.
        src
            This property is of type String.

    The WebCGMMetafile object has the following methods:
```

addEventListener(type,listener);
The type parameter is of type String.
The listener parameter is a WebCGMEventListener object.
removeEventListener(type,listener);
The type parameter is of type String.
The listener parameter is a WebCGMEventListener object.
setRedraw(value);
The value parameter is of type String.

Prototype Object WebCGMNode
The WebCGMNode class has the following constants:
WebCGMNode.PICTURE_NODE
This constant is of type Number and its value is 1.
WebCGMNode.APP_STRUCTURE_NODE
This constant is of type Number and its value is 2.
WebCGMNode.XML_METADATA_NODE
This constant is of type Number and its value is 3.
WebCGMNode.TEXT_NODE
This constant is of type Number and its value is 4.
WebCGMNode.ATTR_NODE
This constant is of type Number and its value is 5.

Object WebCGMNode
The WebCGMNode object has the following properties:
nodeName
This read-only property is of type String.
nodeValue
This read-only property is of type String, can raise a WebCGMException object on retrieval.
nodeType
This read-only property is of type Number.
parentNode
This read-only property is a WebCGMNode object.
childNodes
This read-only property is a WebCGMNodeList object.
firstChild
This read-only property is a WebCGMNode object.
lastChild
This read-only property is a WebCGMNode object.
previousSibling
This read-only property is a WebCGMNode object.
nextSibling
This read-only property is a WebCGMNode object.
attributes
This read-only property is a WebCGMNodeList object.
ownerPicture
This read-only property is a WebCGMPicture object.
namespaceIRI
This read-only property is of type String.
prefix
This read-only property is of type String.
localName
This read-only property is of type String.

The WebCGMNode object has the following methods:
hasChildNodes()
This method returns a Boolean.
hasAttributes()

        This method returns a Boolean.
    getAttributeNS(namespaceIRI, localName)
        This method returns a String.
        The namespaceIRI parameter is of type String.
        The localName parameter is of type String.
    setAttributeNS(namespaceIRI, qualifiedName, value)
        This method has no return value.
        The namespaceIRI parameter is of type String.
        The qualifiedName parameter is of type String.
        The value parameter is of type String.
    getElementsByTagNameNS(namespaceIRI, localName)
        This method returns a WebCGMNodeList object.
    The namespaceIRI parameter is of type String.
        The localName parameter is of type String.

Object WebCGMPicture
    WebCGMPicture has all the properties and methods of the WebCGMNode object as well as the properties and methods defined below.

    The WebCGMPicture object has the following properties:
        width
            This read-only property is of type Number.
        height
            This read-only property is of type Number.
        pictid
            This read-only property is of type String.

    The WebCGMPicture object has the following methods:
        applyCompanionFile(fileIRI)
            This method returns a Boolean.
            The fileIRI parameter is of type String.
        getAppStructureById(apsId)
            This method retuns a WebCGMAppStructure.
            The apsId parameter is of type String.
        getAppStructuresByName(apsName)
            This method retuns a WebCGMNodeList object.
            The apsName parameter is of type String.
        highlight(nodes,type)
            This method has no return value.
            The nodes parameter is a WebCGMNodeList object.
            The type parameter is of type WebCGMString.
        clearHighlight()
            This method has no return value.
            This method has no parameters.
        setPictureVisibility(visibility)
            This method has no return value.
            The visibility parameter is of type String.
        setStyleProperty(style,value)
            This method has no return value.
            The style parameter is of type String.
            The value parameter is of type String.
        reloadPicture()
            This method has no return value.
        getStyleProperty(style)
            This method returns a String.
            The style parameter is of type String.
        setView(view)
            This method has no return value.

The view parameter is of type WebCGMRect.

Object WebCGMAppStructure

WebCGMAppStructure has all the properties and methods of the WebCGMNode object as well as the properties and methods defined below.

The WebCGMAppStructure object has the following properties:

apsId

This read-only property is of type String.

nameCount

This read-only property is of type Number.

linkuriCount

This read-only property is of type Number.

The WebCGMAppStructure object has the following methods:

getAppStructureAttr(name)

This method returns a String.
The name parameter is of type String.

setAppStructureAttr(name,value)

This method has no return value.
The name parameter is of type String.
The value parameter is of type String.

removeAppStructureAttr(name)

This method has no return value.
The name parameter is of type String.

setStyleProperty(style,value)

This method has no return value.
The style parameter is of type String.
The value parameter is of type String.

toNodeList()

This method has no parameters.
This method returns a WebCGMNodeList object.

getObjectExtent()

This method has no parameters.
This method returns a WebCGMRect.

getStyleProperty(style)

This method returns a String.
The style parameter is of type String.

translate(dx,dy,replace)

This method has no return value.
The dx parameter is of type Number.
The dy parameter is of type Number.
The replace parameter is of type String.

rotate(angle,cx,cy,replace)

This method has no return value.
The angle parameter is of type Number.
The cx parameter is of type Number.
The cy parameter is of type Number.
The replace parameter is of type String.

scale(sx,sy,replace)

This method has no return value.
The sx parameter is of type Number.
The sy parameter is of type Number.
The replace parameter is of type String.

setTransform(matrix,replace)

This method has no return value.
The matrix parameter is of type WebCGMMatrix.

The replace parameter is of type String.
getTransform(type)
The type parameter is of type String.
This method returns a WebCGMMatrix.

Object WebCGMNodeList
The WebCGMNodeList object has the following properties:
count
This read-only property is of type Number.

The WebCGMNodeList object has the following methods:
item(index)
This method returns a WebCGMNode object.
The index parameter is of type Number.
removeItem(index) // Raises WebCGMException when the list cannot be modified.
This method returns a WebCGMNode object.
The index parameter is of type Number.
appendItem(newItem) // Raises WebCGMException when the list cannot be modified.
This method returns a WebCGMNode object.
The newItem parameter is a WebCGMNode object.

Object WebCGMAttr
WebCGMAttr has all the properties and methods of the WebCGMNode object as well as the properties and methods defined below.

The WebCGMAttr object has the following properties:
name
This read-only property is of type String.
value
This property is of type String.
ownerNode
This read-only property is a WebCGMNode object.

Object WebCGMEventListener
This is an ECMAScript function reference. This method has no return value. The parameter is a WebCGMEvent object.

Object WebCGMEvent
The WebCGMEvent object has the following properties:
type
This read-only property is of type String.
target
This read-only property is a WebCGMNode object.
button
This read-only property is of type Number.
numPressed
This read-only property is of type Number.
clientX
This read-only property is of type Number.
clientY
This read-only property is of type Number.
ctrlKey
This read-only property is of type Boolean.
shiftKey
This read-only property is of type Boolean.
altKey
This read-only property is of type Boolean.

metaKey
This read-only property is of type Boolean.

The WebCGMEvent object has the following methods:
preventDefault()
This method has no return value.

Back to top of chapter

---

9. Application Configurable Items

This chapter and its sections are normative, unless otherwise indicated.

Contents

9.1 Introduction

This section is informative (non-normative).

9.1.1 Scope and purpose

The WebCGM Aplication Configuration Items file (ACI) allows the user to improve font interchange by specifying a desired font mapping between when WebCGM applications process WebCGM content.

The ACI file also provides a mechanism to specify default handling for certain CGM Version 3 elements that may not have a specific default treatment according to the rules of CGM:1999 or the WebCGM profile.

The ACI file is designed as a set of initialization directives for a viewer, and thus is intended to be associated with a viewer instance and processed once per viewer invocation. The normative specification of methods to associate an ACI file with a viewer is beyond the scope of this version of WebCGM. Some recommedations are given in this chapter.

9.1.2 Motivation

The use cases and requirements for the default-setting functionality of the ACI include several scenarios that the ACI file should satisfy:

1. In Version 1 or Version 2 CGM files, it is not possible to control things like LINE CAP, which is a Version 3 CGM element. Viewers have to select a treatment at random. There is desire to be able to specify consistent and uniform treatment.
2. In V3/V4 files, the CGM:1999-specified default for things like LINE CAP is: 1, "unspecified". CGM:1999 did this so that behavior of viewers would be backward compatible. I.e., the rendering of an otherwise identical file would not change purely based on "V1" versus "V3" in the metafile version. There is desire to be able to specify a consistent and uniform default.
3. The CGM line types 1..5 (solid, dash, dot, dash-dot, dash-dot-dot) are generic in the sense that they only need to be recognizably per the description (e.g., dash-dot). Similarly for the 6 generic hatch styles. There is desire to be able to specify consistent and uniform treatment.

9.2 ACI content and conformance

9.2.1 File overview

The WebCGM Application Configurable Item (ACI) file is an XML instance specifying default values for various CGM Version 3 attribute and control elements, font substitutions to be performed by WebCGM processors . It is

made up of a root element (<webcgmConfig>) followed by an optional font map (<fontMap>) elment and an optional default style properties (<defaultAttributes>) element.

9.2.2 Conformance of ACI files

A file is a conforming WebCGM 2.1 ACI document if it adheres to the specifications described in this (WebCGM 2.1) document, including those in the WebCGM 2.1 ACI DTD, and in addition:

- it is a well-formed XML document (according to XML 1.0 [XML10]);
- its root element is webcgmConfig

9.2.3 Association of ACI files with viewers

This subsection is informative (non-normative).

Methods to associate an ACI file with a viewer — being operating system, viewer, and application dependent — are considered to be beyond the scope of the normative specifications of this version of WebCGM. To improve interoperability between the various providers of WebCGM 2.1 viewers, the following recommendations for associating an ACI file with a WebCGM 2.1 viewer are included.

It is recommended ACI file have the extension ".aci" (all lowercase) on all platforms.

To convey to a WebCGM 2.1 viewer the location of an ACI file, an environment variable name WebCGM_ACI_File is defined. This environment variable could be set at the time of the viewer installation or modified by a user. On Windows this should be System environment variable that would apply to all users.

Example:

WebCGM_ACI_File=c:/Documents and Settings/All Users/Application Data/CGM Open/WebCGM

If not feasible for a viewer to set an environment variable during viewer installation or users are prohibited from doing so due to security policies that are becoming commonplace in large organizations, then the viewer should provide an alternative method for users to specify the ACI file search path, for example in a viewer preferences or configuration menu.

9.3 ACI elements

9.3.1 The 'webcgmConfig' element

An ACI file must have webcgmConfig as its root element.

```
<!ELEMENT webcgmConfig ( fontMap?, defaultAttributes? ) >
```

9.3.2 The 'fontMap' element

The `fontMap` element is the ACI mechanism for font substitution.

```
<!ELEMENT fontMap ( defaultFont?, mapList* ) >
```

9.3.2.1 The 'defaultFont' element

The `defaultFont` element specifies, via its `useFont` attribute, a string defining the font to be used when a requested mapping cannot be accomplished.

```
<!ELEMENT defaultFont EMPTY >
<!ATTLIST defaultFont
          useFont CDATA #REQUIRED >
```

Attribute definitions:
useFont="CDATA"
The default font to use when the specified font mapping cannot be accomplished. If the font(s) of `mapList` (`substitutionList`) are not available, and if `defaultFont` (`useFont`) is not available, then the `fontMap` element has no effect, and any viewer fallback action is viewer dependent. The syntax and normalization rules of `useFont` are the same as those of the `substitutionList` attribute, except that `useFont` shall have exactly one font-family name or generic name in its list.

9.3.2.2 The 'mapList' element

The `mapList` element specifies, via its `cgmFont` and `substitutionList` attributes, the font mapping to be performed before rendering of the image. Subject to the value of the `forceSubstitution` attribute (see below), if the metafile uses a font whose name matches `cgmFont` (after normalization of both font names as described below), then the viewer shall substitute the first available font in the priority-ordered list of the `substitutionList` attribute.

```
<!ELEMENT maplist EMPTY >
<!ATTLIST maplist
          forceSubstitution ( yes | no ) “yes”
          cgmFont CDATA #REQUIRED
          substitutionList CDATA #REQUIRED >
```

Attribute definitions:
forceSubstitution= { yes | no }
If 'yes', then the subsitution is unconditional -- it is attempted regardless of whether or not the viewer has the font specified by `cgmFont` available. If 'no', then the substitution is conditional -- the viewer only attempts the substitution if it does not have the font specified by `cgmFont` available.
cgmFont="CDATA"
The name of the font in the metafile for which font substitution is requested. Before attempting to match a font used in the metafile to the string `cgmFont`, the both font names are normalized by: converting to lower-case; and stripping out all whitespace, UNDERSCORE, and HYPHEN characters. Note: These normalization rules are applicable for font names specified using the characters of ISOLatin1. They will likely be inapplicable for font names specified using other non-Latin characters.
substitutionList="CDATA"
A comma-separated, priority-ordered list of comprised of font-family names (e.g., Arial) and generic names (e.g., san-serif). The processor shall use the name in the `substitutionList` that it has available. The list syntax and normalization are derived from the specifications of CSS 2.0 [CSS20]. In particular, the values and syntax of the substitutionList attribute are derived from CSS's definition of the font-family property:

- the names in the list may be an actual font-family name, or one of the five generic families: 'serif', 'sans-serif', 'cursive', 'fantasy', and 'monospace';
- if the names contain whitespace characters, they must be quoted;
- in the absence of quotes, whitespace is normalized by stripping leading and trailing whitespace characters and compressing internal whitespace strings to a single whitespace character.

Example:

<maplist cgmFont="helvetica" substitutionList="Arial, 'FontCorp Swiss', sans-serif">

9.3.3 The 'defaultAttributes' element

The `defaultAttributes` element is the ACI mechanism to specify default values for certain CGM attribute and control elements whose default is otherwise under-specified in CGM:1999 and the WebCGM profile. This element, when processed at initialization time, will set default values for the applicable metafile elements. If the contents of the metafile, upon interpretation, explicitly set the applicable metafile element (either via METAFILE DEFAULTS REPLACEMENT or via the element itself in the body of the picture), then that explicit intra-metafile setting supersedes the `defaultAttributes` setting

```
<!ELEMENT defaultAttributes ( lineCap | edgeCap | lineJoin | edgeJoin
                            | lineTypeCont | edgeTypeCont | mitreLimit
                            | restrTextType | lineEdgeTypeDef | hatchStyleDef )+ >
```

It is possible to define the default values of the following Version 3 elements:

- Line Cap
- Edge Cap
- Line Join
- Edge Join
- Line Type Continuation
- Edge Type Continuation
- Mitre Limit
- Restricted Text Type
- Line and Edge Type Definition
- Hatch Style Definition

The parameter definition for each element is consistent with its ISO/IEC 8632:1999 CGM parameter definition.

9.3.3.1 The 'lineCap' element

The `lineCap` element is an EMPTY element that specifies the desired rendering of the end caps of lines and of dashes within lines.

```
<!ELEMENT lineCap EMPTY >
<!ATTLIST lineCap
          lineCapInd ( 1 | 2 | 3 | 4 ) "1"
          lineDashInd ( 1 | 2 | 3 ) "1"
>
```

Attribute definitions:
lineCapInd="1|2|3|4|5"
The line cap indicator is restricted to value of 1-4. (Note that the ISO/IEC 8632:1999 value "5" is disallowed in the WebCGM profile.)

- 1 - unspecified
- 2 - butt
- 3 - round
- 4 - projecting square

The default value is '1' or 'unspecified'.

lineDashInd="1|2|3"

The line dash cap indicator is restricted to values of 1-3 as defined in ISO/IEC 8632:1999.

- 1 - unspecified
- 2 - butt
- 3 - match

The default value is '1' or 'unspecified'.

9.3.3.2 The 'edgeCap' element

The `edgeCap` element is an EMPTY element that specifies the desired rendering of the end caps of edges and of dashes within edges.

```
<!ELEMENT edgeCap EMPTY >
<!ATTLIST edgeCap
          edgeCapInd ( 1 | 2 | 3 | 4 ) "1"
          edgeDashInd ( 1 | 2 | 3 ) "1"
>
```

Attribute definitions:
edgeCapInd="1|2|3|4"
The edge cap indicator is restricted to value of 1-4. (Note that the ISO/IEC 8632:1999 value "5" is disallowed in the WebCGM profile.)

- 1 - unspecified
- 2 - butt
- 3 - round
- 4 - projecting square

The default value is '1' or 'unspecified'.

edgeDashInd="1|2|3"
The edge dash cap indicator is restricted to values of 1-3 as defined in ISO/IEC 8632:1999.

- 1 - unspecified
- 2 - butt
- 3 - match

The default value is '1' or 'unspecified'.

9.3.3.3 The 'lineJoin' element

The `lineJoin` element is an EMPTY element that specifies the desired rendering of the join characteristics at vertices of the line element.

```
<!ELEMENT lineJoin EMPTY >
<!ATTLIST lineJoin
          lineJoinInd ( 1 | 2 | 3 | 4 ) "1"
>
```

Attribute definitions:
lineJoinInd="1|2|3|4"
The line join indicator is restricted to value of 1-4 as defined in ISO/IEC 8632:1999.

- 1 - unspecified
- 2 - mitre
- 3 - round
- 4 - bevel

The default value is '1' or 'unspecified'.

9.3.3.4 The 'edgeJoin' element

The `edgeJoin` element is an EMPTY element that specifies the desired rendering of the join characteristics at vertices of the edge element.

```
<!ELEMENT edgeJoin EMPTY >
<!ATTLIST edgeJoin
          lineJoinInd ( 1 | 2 | 3 | 4 ) "1"
>
```

Attribute definitions:
edgeJoinInd="1|2|3|4"
The edge join indicator is restricted to value of 1-4 as defined in ISO/IEC 8632:1999.

- 1 - unspecified
- 2 - mitre
- 3 - round
- 4 - bevel

The default value is '1' or 'unspecified'.

9.3.3.5 The 'lineTypeCont' element

The `lineTypeCont` element of the ACI is an EMPTY element that specifies the desired rendering of the line type continuation characteristics of the line element.

```
<!ATTLIST lineTypeCont
          lineContMode ( 1 | 2 | 3 | 4 ) "1"
>
```

Attribute definitions:
lineContMode="1|2|3|4"
The line type continuation indicator is restricted to value of 1-4 as defined in ISO/IEC 8632:1999.

- 1 - unspecified
- 2 - continue
- 3 - restart
- 4 - adaptive continue

The default value is '1' or 'unspecified'.

9.3.3.6 The 'edgeTypeCont' element

The `edgeTypeCont` element is an EMPTY element that specifies the desired rendering of the edge type continuation characteristics of the edge element.

```
<!ATTLIST edgeTypeCont
          edgeContMode ( 1 | 2 | 3 | 4 ) "1"
>
```

Attribute definitions:
edgeContMode="1|2|3|4"
The edge type continuation indicatoris restricted to value of 1-4 as defined in ISO/IEC 8632:1999.

- 1 - unspecified
- 2 - continue
- 3 - restart
- 4 - adaptive continue

The default value is '1' or 'unspecified'.

9.3.3.7 The 'mitreLimit' element

The `mitreLimit` element is an EMPTY element that specifies how line join at vertices are achieved.

```
<!ELEMENT mitreLimit EMPTY >
<!ATTLIST mitreLimit
          limitVal CDATA #REQUIRED
>
```

Attribute definitions:
limitVal="CDATA"
The mitre limit value is defined as a scale factor applied to current line and edge width. Valid values are non-negative real numbers.

9.3.3.8 The 'restrTextType' element

The `lineCap` element is an EMPTY element that specifies the desired rendering of the end caps of lines and dashes within lines.

```
<!ELEMENT restrTextType EMPTY >
<!ATTLIST restrTextType
          restrType ( 1 | 2 | 3 | 4 | 5 | 6 ) "1"
>
```

Attribute definitions:
restrType="1|2|3|4|5|6"
The restricted text type is restricted to value of 1-6 as defined in ISO/IEC 8632:1999.

- 1 - basic
- 2 - boxed-cap
- 3 - boxed-all
- 4 - isotropic-cap
- 5 - isotropic-all
- 6 - justified

The default value is '1' or 'basic'.

9.3.3.9 The 'lineEdgeTypeDef' element

The `lineEdgeTypeDef` element is an element that specifies the desired rendering of the standard line type in ISO/IEC 8632:1999.

```
<!ELEMENT lineEdgeTypeDef ( dashLength+ ) >
<!ATTLIST lineEdgeTypeDef
          lineIndex ( 1 | 2 | 3 | 4 | 5 ) #REQUIRED
          repeatLength CDATA #REQUIRED
>
```

Attribute definitions:
lineIndex="1|2|3|4|5"
The line type index is restricted to value of 1-5 as defined in ISO/IEC 8632:1999.

- 1 - solid
- 2 - dash
- 3 - dot
- 4 - dash-dot
- 5 - dash-dot-dot

repeatLength="CDATA"
The dash cycle repeat length defines the length of one complete repetition of the dash pattern in NVDC units.

9.3.3.9.1 The 'dashLength' element

The `dashLength` element is a string, in the encoding of the ACI file, that contains a list of non-negative integers in the format of the WebCGMString List-of-number subtype. The integers specify the lengths of each dash and gap in the defined line pattern in abstract units, that are then normalized as a whole pattern to the `repeatLength` attribute of `lineEdgeTypeDef`. The first integer corresponds to solid, the second to gap, the third to solid, etc.

```
<!ELEMENT dashLength ( #PCDATA ) >
```

9.3.3.10 The 'hatchStyleDef' element

The `hatchStyleDef` element specifies the desired rendering of the standard hatches in ISO/IEC 8632:1999.

```
<!ELEMENT hatchStyleDef ( directionVectors, ( gapWidth+ ), ( lineTypeIndex+ ) ) >
<!ATTLIST hatchStyleDef
          hatchIndex ( 1 | 2 | 3 | 4 | 5 | 6 ) #REQUIRED
          styleInd ( parallel | crosshatch ) #REQUIRED
          cycleLength CDATA #REQUIRED
          numberOfLines CDATA #REQUIRED
>
```

Attribute definitions:
hatchIndex="1|2|3|4|5|6"
The hatch style index is restricted to value of 1-6 as defined in ISO/IEC 8632:1999.

- 1 - horizontal equally spaced parallel lines
- 2 - vertical equally spaced parallel lines
- 3 - positive slope equally spaced parallel lines
- 4 - negative slope equally spaced parallel lines
- 5 - horizontal/vertical crosshatch
- 6 - positive slope/negative slope crosshatch

styleInd="paralel|crosshatch"
The style indicator indicates whether the hatch pattern is made up of parallel lines or crosshatch lines.

cycleLength="CDATA"
The duty cycle length defines the length of one one complete repetition of the hatch pattern. It is measured perpendicular to the hatch lines and is in NVDC units.

numberOfLines="CDATA"

The number of hatch lines defines the number of entries in the list of gap widths and list of lines types.

9.3.3.10.1 The 'directionVectors', 'vectorX' and 'vectorY' elements

The `directionVectors` element contains the `vectorX` and `vectorY` elements to define the direction of the hatch lines. The units are NVDC. If the style indicator is parallel, only the first set of vectors is used, but both must be present. The 4 numbers of `directionVectors` are encoded in the format of the WebCGMString List-of-number subtype.

```
<!ELEMENT directionVectors ( vectorX, vectorY, vectorX, vectorY ) >
<!ELEMENT vectorX ( #PCDATA ) >
<!ELEMENT vectorY ( #PCDATA ) >
```

9.3.3.10.2 The 'gapWidth' element

The `gapWidth` element defines the width between the lines in the hatch pattern. It is a list of positive integers. The number of gap width integers cooresponds to the number-of-lines attribute in the hatch style definition. The list of values of `gapWidth` are encoded in the format of the WebCGMString List-of-number subtype.

```
<!ELEMENT gapWidth ( #PCDATA ) >
```

9.3.3.10.3 The 'lineTypeIndex' element

The `lineTypeIndex` element defines the line types of the lines that make up the hatch pattern. The number of line type index elements cooresponds to the number of lines attribute in the hatch style definition. Valid values are 1-5. The list of values of `lineTypeIndex` are encoded in the format of the WebCGMString List-of-number subtype.

```
<!ELEMENT lineTypeIndex ( #PCDATA ) >
```

9.4 The complete ACI DTD

The complete WebCGM Application Configurable Items (ACI) file DTD follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ============================================================= -->
<!-- This is the WebCGM Application Configurable Item file DTD     -->
<!-- for use with  WebCGM 2.1                                      -->
<!-- ============================================================= -->
<!-- Original issue: March 2008                                    -->
<!--                                                               -->
<!-- Revision history:                                             -->
<!--      June 2008 - modified for changes to CD02 ACI text.       -->
<!--                                                               -->
<!-- ============================================================= -->
<!--                                                               -->
<!ELEMENT webcgmConfig ( fontMap?, defaultAttributes? )              >

<!ELEMENT fontMap ( defaultFont?, maplist* )  >

<!ELEMENT defaultFont EMPTY                                          >
<!ATTLIST defaultFont
          useFont CDATA #REQUIRED >

<!ELEMENT maplist EMPTY                                              >
<!ATTLIST maplist
          forceSubstitution ( yes | no ) "yes"
          cgmFont CDATA #REQUIRED
          substitutionList CDATA #REQUIRED                           >

<!ELEMENT defaultStyleProp ( lineCap | edgeCap | lineJoin | edgeJoin
                           | lineTypeCont | edgeTypeCont | mitreLimit
                           | restrTextType | lineEdgeTypeDef
                           | hatchStyleDef )+                        >

<!ELEMENT lineCap EMPTY                                              >
<!ATTLIST lineCap
```

```
          lineCapInd ( 1 | 2 | 3 | 4 | 5 ) "1"
          lineDashInd ( 1 | 2 | 3 ) "1"                                    >

<!ELEMENT edgeCap EMPTY                                                    >
<!ATTLIST edgeCap
          edgeCapInd ( 1 | 2 | 3 | 4 | 5 ) "1"
          edgeDashInd ( 1 | 2 | 3 ) "1"                                    >

<!ELEMENT lineJoin EMPTY                                                   >
<!ATTLIST lineJoin
          lineJoinInd ( 1 | 2 | 3 | 4 ) "1"                                >

<!ELEMENT edgeJoin EMPTY                                                   >
<!ATTLIST edgeJoin
          edgeJoinInd ( 1 | 2 | 3 | 4 ) "1"                                >

<!ELEMENT lineTypeCont EMPTY                                               >
<!ATTLIST lineTypeCont
          lineContMode ( 1 | 2 | 3 | 4 ) "1"                               >

<!ELEMENT edgeTypeCont EMPTY                                               >
<!ATTLIST edgeTypeCont
          edgeContMode ( 1 | 2 | 3 | 4 ) "1"                               >

<!ELEMENT mitreLimit EMPTY                                                 >
<!ATTLIST mitreLimit
          limitVal CDATA #REQUIRED                                         >

<!ELEMENT restrTextType EMPTY                                              >
<!ATTLIST restrTextType
          restrType ( 1 | 2 | 3 | 4 | 5 | 6 ) "1"                          >

<!ELEMENT lineEdgeTypeDef ( dashLength+ )                                  >
<!ATTLIST lineEdgeTypeDef
          lineIndex ( 1 | 2 | 3 | 4 | 5 ) #REQUIRED
          repeatLength CDATA #REQUIRED                                     >

<!ELEMENT dashLength ( #PCDATA )                                           >

<!ELEMENT hatchStyleDef ( directionVectors, ( gapWidth+ ),
                                            ( lineTypeIndex+ ) )           >
<!ATTLIST hatchStyleDef
          hatchIndex ( 1 | 2 | 3 | 4 | 5 | 6 ) #REQUIRED
          styleInd ( parallel | crosshatch ) #REQUIRED
          cycleLength CDATA #REQUIRED
          numberOfLines CDATA #REQUIRED                                    >

<!ELEMENT directionVectors ( vectorX, vectorY, vectorX, vectorY )          >

<!ELEMENT vectorX ( #PCDATA )                                              >
```

```
<!ELEMENT vectorY ( #PCDATA )                                    >

<!ELEMENT gapWidth ( #PCDATA )                                   >

<!ELEMENT lineTypeIndex ( #PCDATA )                              >
```

Back to top of chapter

WebCGM 2.1 — Appendixes

Contents

A. Acknowledgements

In addition to the listed editors of this specification, the following individuals have contributed significantly to the present WebCGM 2.1 specification:

- Forrest Carpenter, System Development Inc.
- Franck Duluc, Airbus
- Stuart Galt, The Boeing Company
- Ulrich Laesche, Ematek Informatik GMBH
- Don Larson, Larson Software Technology
- Robert Orosz, Auto-trol Technology
- Andrew Moorhouse, UK Ministry of Defence
- Dieter Weidenbrueck, ITEDO

Additionally, the following individuals made signficant contributions to predecessor versions, WebCGM 2.0 and WebCGM 1.0:

- Ralf Berger, ITEDO
- Chris Lilley, W3C
- Thierry Michel, W3C
- Felix Sasaki, W3C
- Kentarou Fukuda, IBM
- Al Gilman, W3C Invited Expert
- Kevin O'Kane, Auto-trol Technology

- Maryse Da-Ponte, Airbus
- Bruce Garner, Analyst/Consultant
- Alan Hester, Xerox Corporation
- Bob Hopgood, CCLRC
- Brad Powell, Zeh Graphic Systems
- Dave Rahnis, Bentley Systems
- Lynne Rosenthal, NIST
- John Gebhardt, Intercap Graphics Systems
- Roy Platon, CCLRC
- Ty Bartosh, Jeppesen Inc.

B. What's new in WebCGM 2.1

This section is informative (non-normative).

WebCGM 2.1 builds upon WebCGM 2.0 and adds these features:

1. Geometric transform definition for objects, via DOM and XCF facilities, plus getTransform() inquiry method.
2. getStyleProperty() inquiry method added to DOM.
3. a number of new Style Properties are added to the list of those settable via the setStyleProperty() method.
4. Application configurable items: font mapping; defaults setting.
5. g-zip compression of whole metafiles.
6. new setRedraw() method on WebCGMMetafile allows postponement / control of when redraws happen.
7. new setView() method on WebCGMPicture allows the picture view to be defined via the DOM.
8. new getObjectExtent() method on WebCGMAppStructure facilitates getting the bounding extent of one or more objects, for example to use with new setView()
9. Transparency Clarifications.
10. A number of 2.0-deprecated items were made obsolete for 2.1.

C. Glossary

API, Application Programming Interface
: An Application Programming Interface (API) is a set of functions or methods used to access some functionality.

Application Structure, APS
: The CGM structure for grouping other elements and assigning them a unique identifier, a type, and attaching attributes. WebCGM contains five valid APS types: grobject, layer, para, subpara, grnode.

APS Attribute, Application Structure Attribute
: The CGM element that is used to define and associate an attribute with an APS. WebCGM contains ten valid APS Attribute types: region, viewcontext, linkuri, layername, layerdesc, screentip, name, content, visibility, interactivity.

Cascading profile
: A method by which closely related profiles can be expressed efficiently, by only enumerating the differences between a given profile and a base profile such as WebCGM.

CGM, Computer Graphics Metafile
: ISO/IEC standard 8632:1999 (CGM:1999), CGM is a metafile format consisting of composite raster and scalable vector graphics information.

DOM, Document Object Model
: A Document Object Model is a platform- and language-neutral interface that allows programs and scripts to dynamically access and manipulate the content, structure and style of documents. WebCGM defines a DOM of limited scope, allowing discovery and navigation of the WebCGM structure tree, transient manipulation of styles and standardized metadata, and discovery and manipulation of application-specific metadata. The WebCGM DOM is based upon and borrows from the principles the W3C DOM Level 2 and DOM Level 3 Recommendations.

EBNF, Extended Bauer Normal Form
: A formal specification technique used in WebCGM to express pieces of grammar and syntax such as the IRI fragment syntax.

fragment (IRI fragment)
: As standardized in RFC 3986 and RFC 3987, the fragment is a part of a IRI that is separated from the path/file name by a "#" character, and provides information that is reserved for processing by agents that are invoked by the browser to handle the resource information type of the IRI. WebCGM standardizes the syntax and semantics of such fragments, for transmitting object and picture selection and behavior information to WebCGM viewers.

handler, event handler
: An event handler is a method called by the WebCGM DOM implementation whenever a specified event occurs. Users register the event types they are interested in by calling the addEventListener() method of the WebCGMMetafile object.

host application
: The user agent, typically is a script that controls the behavior of a web page. In the context of WebCGM the host application controls the behavior of the WebCGM renderer through the DOM.

host document
: The web page, contains a combination of both HTML and WebCGM content.

host environment
The combination of the host application and the host document

inheritance
The determination of values of APS Attributes and Style Properties for objects in the WebCGM hierarchical object tree, when all aspects have not been explicitly specified for all objects in the tree. WebCGM defines an inheritance model that is closely based on CSS (Cascading Style Sheets, a W3C Recommendation).

interactive region
the effective geometric region for the purposes of all interactive cursor and mouse operations, such as picking and mouseover. By default, the *drawn graphical primitives* of the object define the interactive region. For filled-area primitives this includes: the edge, if edge visibility is 'on'; the interior, if the interior style is other than 'empty' or 'hollow'; and, the boundary, for interior style 'hollow'. For all graphical primitive types, *drawn graphical primitives* exclude any that are fully transparent (so a fully transparent object is equivalent to an empty object, for purposes of interactive region definition). If the object contains a 'region' APS Attribute, then that region area is the interactive region.

listener, event listener
The event listener specifies the interface via which users register event handlers with the WebCGM DOM implementation. This interface consists of a single method, handleEvent(), which the WebCGM DOM implementation calls whenever specified events occur. Users pass WebCGMEventListener objects as arguments to the addEventListener() method of the WebCGMMetafile object to register a specific event handler with the WebCGM DOM implementation.

metadata
Non graphical information contained within or associated with standard graphical files, in WebCGM metadata supports such ancilliary functions as hierarchical picture structuring, object identification and navigation, and association of application-specific non-graphical data with graphical objects.

metafile
A mechanism for retraining and transporting graphical data and control information, containing a device independent description of one or more pictures.

namespace, NS
A method devised and standardized by the *XML Namespaces* standard, namespace provides a way to distinguish to which specification and grammar elements and attributes belong, when information from distinct XML languages is mixed in the same document. WebCGM uses namespaces to separate standardized WebCGM XCF elements and attributes from embedded and intermingled application-specific metadata.

Normalized Device Coordinates, NVDC
The coordinate system that is used to communicate coordinate data through the WebCGM DOM, NVDC is VDC normalized so that the origin is lower-left and units are millimeters.

object
An Application Structure (grobject, para, subpara) or Picture in a WebCGM.

object behavior
One of a set of (thirteen) ways in which the view of an object or collection of objects is presented following the execution of a hyperlink to the object(s). WebCGM standardizes a number of objects that give complete control over the zoom, pan, and highlight aspects of the view.

picture behavior
One of a set of ways in which the view of a picture (CGM) or document (HTML) is handled following the execution of a hyperlink to the content. Based on the 'target' attribute of the HTML 'a' tag, picture behaviors allow the specification of the new view to occur in a new window, to overwrite the whole contents of the source window, to overwrite the parent window, etc.

PPF, Profile Proforma
A method of expressing profiles standardized in CGM:1999 (clause 9), the PPF presents a profile as a single table covering all aspects of the CGM standard, with a reference column (enumerating a Model Profile or other base profile), and a column to define the target profile by comparison to the reference column.

Style Property
A graphical attributes that may be applied and manipulated at the APS or picture level, transiently, by WebCGM DOM and WebCGM XCF. WebCGM 2.0 defines nine style properties.

target rectangle
A rectangular region around the target object(s) that is used for the application of object behaviors following hyperlink execution. The target region is defined by APS Attributes and/or target geometry, and for example provides the area to be encompassed by a zoomed view.

WDOM, WebCGM DOM
The WebCGM Document Object Model, this terminology is used when it is needed to unambiguously distinguish the WebCGM DOM from the generalized W3C DOM specifications.

XCF, XML Companion File
An XML file format defined by WebCGM, that can be used to externalize metadata from WebCGM instances and bind it to objects in the metafile.

D. Change log

This section is informative (non-normative).

Note: The following subsections list all significant changes applied to the WebCGM 2.0 specification to derive this draft of the WebCGM 2.1 specification.

D.1 Changes from CD01 to CD02 of WebCGM 2.1

---20090527 draft (CD03) includes all of below changes---

- Corrections to ACI external dtd file (webcgmConfig21.dtd) exactly matching those in 2nd LCWD draft.

---20080625 draft (CD02) includes all of below changes---

- Update ECMAScript (Ch.8) to align with Ch.5 DOM changes: WebCGMRect, WebCGMMatrix, float datatypes replacing some WebCGMString.
- Improve specifications of parameter in Chapter 9 with explicit "WebCGMString List-of-number subtype" statements.
- Added non-normative "9.2.3 Association of ACI files with viewers" to ACI chapter.
- Rules in 3.2.1.7 that CGM:1999 layout rules take precedence over DOM geometric transform ("no effect") on substring APS
- Update section 9.4 (complete dtd of the ACI) and external DTD file webcgmConfig21.dtd.
- Update section 4.4 (complete dtd of the XCF) and external DTD file webcgm21.dtd.
- Integrate executable example into Regex Example (Appendix F).
- Throughout Chapter 4 (XCF), change "geometric transform style properties" to "geometric transform", and "geometricTransformSP" to "geometricTransform".
- In Chapter 4 (XCF) geometricTransform entity definition, change "matrix" to "setTransform".
- Adjust 4.3.1 for new Float and WebCGMMatrix parameter types in DOM transform attributes and parameters (old DOM types were WebCGMString).
- Fixed examples in 4.3.1 so that coordinates are proper NVDC (0.0-1.0).
- Clarify in 3.1.2.7 that RT/AT layout rules of CGM:1999 have precedence over possible substring-APS geometric transform.

---20080617 editor's draft includes all of below changes---

- (Partially) implemented Needham agreements on ACI font-sub capability.
- (Partially) implemented Needham agreements on ACI defaultAttributes capability.
- Implemented editorial and other corrections from Chapter 9 partial review.
- Corrected error in T.20.22 of Section 6.9 -- 'geometric pattern' is not a valid interior style.
- Improve the uniformity of style at the start of all Chapters.
- Implement normative reference comments in Ch.1 (postpone Unicode/ISO10646 and W3C-related updates pending further expert input)
- ACI-file-reference is removed from the fragment grammar, in section 3.1.1 and throughout Chapter 3.
- Retitle, " 5.7.1 Common definitions", with former 5.7.1 WebCGMException becoming 5.7.1.1 Exception WebCGMException
- Added new 5.7.1.2 Interface WebCGMRect; modified getObjectExtent() (section 5.7.6) and setView() (section 5.7.5) to use WebCGMRect.
- Added new 5.7.1.3 Interface WebCGMMatrix; modified getTransform() and setTransform() [formerly matrix()] accordingly on WebCGMAppStructure interface.
- Change section 6.18 JPEG to informative (it is redundant with the normative JPEG entry in the Graphical Register).
- Normalize the names and wording of new Style Properties in DOM WebCGMpicture interface (section 5.7.5).
- Normalize the names and wording of new Style Properties in XCF section 4.3.2 and section 4.4, and in the external DTD file.
- Remove ACI Processing Instruction example (4.4) from Section 4.1.
- Placeholder message in 3.2.1.7 about CGM:1999 RT/AT layout rules versus DOM manipulation on partial text elements.
- Add clarifying wording about 'grnode' permissibility to section 5.7.4, the WebCGMNode interface.
- Add detailed clarification of the effect (none) of invalid 'grnode' as the APS type in WebCGMAppStructure interface (5.7.6).
- In the WebCGMPicture interface, add clarification that 'grnode' returns null in getApsStructureById() method, and is invalid ("no effect") in the node list of highlight().
- Add note to section 3.1.2.5 'grnode' about general exclusion from most DOM functionality.
- Clarify that 'grnode' is not valid in bindById element.

---20080515 editor's draft includes all of below changes---

- General except-grnode wording in 5.7.1 Common definitions(new 5.7.1.4)?
- Fix wording of 'grnode' exception in section 5.3, item #2 in "processing child elements" list.
- Split 7.2.1 Obsolete features into two subsections, for 2.0-obsolete and 2.1-obsolete.
- Move compression types, 'viewport' param, and 3-object-behaviors from "deprecated" to 2.1-obsolete.
- Add deprecation message to `background` param element in section 3.4; add bullet to section 7.2.2 Deprecated features; remove last paragraph of section 2.2.3.
- In WebCGMPicture and ECMAScript binding, changed method name setViewPort() to setView(), parameter type from WebCGMString to four floats.
- Implemented many editorial changes from editorial review of Ch.1 Introduction.
- Implemented many editorial changes from editorial review of Ch.2 Concepts.

D.2 Changes from WebCGM 2.0 to CD01 of WebCGM 2.1

---20080327 draft (CD01) includes all of below changes---

- (CD01 added Ch.9 ACI file format, aci access from frag syntax in Ch.3, aci-processing PI in Ch.4).
- (CD01 added new Style Properties to Ch.5 and Ch.4).
- Added geometric transform methods and getTransformSP to ECMAScript.
- Added getTransformSP() method to WebCGMAppStructure interface.
- Added entity geometricTransformSP to webcgm21.dtd.
- Added geometric transform SP to XCF, as XML attributes, defining entity geometricTransformSP; description and order caveat in sec. 4.3.1.
- Added more normative explanation and supporting material to "Geometric transform Style Property"
- Added scale, translate, rotate, and matrix methods to WebCGMAppStructure interface.
- Added getStyleProperty() inquiry method to Picture interface and WebCGMAppStructure interface.
- Added getObjectExtent() to WebCGMAppStructure interface.
- Added setViewPort() to Picture interface
- Added new style properties to sec. 4.3.2 and sec.4.4, along with sec 5.7.5 and the XCF DTD.
- Added GZIP requirement to Ch.6, Profile, T.13.1.
- Added new Ch.9, "Application Configurable Items", fixed cover page TOC, fixed detailed TOC, added description to end of sec. 2.5.2 and 2.5.4.
- Added g-zip (RFC1952) to normative references, added informative paragraph to 2.4 Encodings.
- Implemented placeholder 5.4.4 geometric transform description.
- General makeover 2.0-to-2.1 of Chapter 5, Document Object Model.
- General makeover 2.0-to-2.1 of Chapter 4, XML Companion File .
- General makeover 2.0-to-2.1 of Chapter 8, ECMAScript binding .
- added setRedraw() method to WebCGMMetafile interface, and to the Ch.8 ECMAScript binding.
- added HTML script code for regular expression searching to new Appendix F and link to it from "Examples" in 5.7.6.
- added getObjectExtent() method to 5.7.6 Interface WebCGMAppStructure and the Ch.8 ECMAScript binding.
- General makeover 2.0-to-2.1 of Chapter 3, Intelligent Content.
- General makeover 2.0-to-2.1 of Chapter 2, Concepts.
- General makeover 2.0-to-2.1 of Chapter 1, Introduction.
- Integrated all 2.0 Errata (as of 2008-03-03).
- General makeover 2.0-to-2.1 of Ch.6 PPF.
- Implemented transparency clarifications: new text in 2.2.3, reference 2.2.3 from 3.4 'background', add informative notes to T.18.3, T.18.4, T.18.16, and T.21.1.
- For sub-string hot spot clarification, modified text in 3.2.1.1, 3.2.1.3, 3.2.1.4, and added new 3.2.1.7.
- Added ISO/IEC 8632-1:1999/Cor 2:2007 (sub-string hotspot clarification) to 1.2 Normative references.
- Revised Appendixes for general 2.0-to-2.1 makeover, restart "D. Change log", restart "What's new in WebCGM 2.1".
- Revised Ch.7 Conformance clause so that it's appropriate for 2.1, including:
  - general 2.0-to-2.1 makeover;
  - fixed test suite references in section 7.6
  - preliminary rework of deprecation-obsoletion scheme for 2.0-to-2.1
- Remade document structure and file naming for WebCGM 2.1.

E. WebCGM accessibility

This section and its subsections are informative (non-normative).

E.1 Introduction

Topics such as internationalization and accessibility have been addressed by W3C in other Recommendations available on the Technical Reports page. The relationship between the specifications of technology modules, such as this one, and those cross-cutting recommendations is discussed in the Specification Guidelines. As suggested in the Specification Guidelines, this section identifies some relationships between the capabilities afforded by WebCGM features and accessibility requirements established by other Recommendations.

Although a WebCGM metafile is a binary file format, it has systematic grouping and annotation features that foster accessibility of WebCGM metafile content. For example, graphical text is ideally stored as Unicode text strings within the metafile, but to handle cases where graphical text images are actually the result of other vector (e.g. stroking) or raster (e.g., bitmaps) graphics in the metafile, WebCGM has the attribute 'content' on the para and subpara objects, that gives the text-string equivalent of the rendered graphical text.

Features that are in the binary metafile itself will, unlike clear-text formats such as HTML, XML, and SVG, require the cooperation and intervention of the WebCGM authoring tools and viewers. As described in [Essential Components of Web Accessibility], several components such as authoring tools, media viewers and developers, have to work together to improve Web accessibility. Therefore, the discussions in the following sections are recommendations to content developers, and builders of authoring tools and viewers.

E.2 Navigation

By Guideline 9 "Provide navigation mechanisms" of [UAAG10] a WebCGM viewer is expected to let users interact with 'enabled' and significant objects in the image. 'Enabled' objects are those which accept user input, such as on screen buttons. By the structure of WebCGM, each APS should be treated as a significant object and be reachable by navigation techniques. By Guideline 1 "Support input and output device independence" the reach of keyboard-actuated navigation should cover this whole set of navigation destinations.

Some notion of forward and backward motion among peer nodes in the WebCGM image should be provided. This should by default move among paragraphs and sub-paragraphs in the order in which they appear in the metafile. The creators of WebCGM instances should ensure that this results in a sensible reading order. However, efficient motion as called for in [UAAG10] Checkpoint 9.9, is not likely to result from one global list or loop of all the plausible navigation destinations. Exploiting the structure of the metafile, structured navigation could take hierarchical or categorical forms. In hierarchical navigation, forward and back motion moves among peer nodes at the same level in the layers-and-objects nesting tree. In categorical navigation, the sequential navigation could exhibit navigation modes which visit only 'grobject' nodes, or only the 'gropbject' nodes with a common 'name.' An example of hierarchical navigation is provided by the player behavior for the [DAISY] standard digital talking book. An example of categorical structured navigation is provided by the diverse navigation modes of the Opera browser. The creators of WebCGM instances should ensure that the layers-and-objects nesting forms a plausible table of contents as annotated with the textual properties (see E.3 below) of the affected nodes, and that collecting nodes of like 'name' forms meaningful slices of what is in the scene.

In this version of WebCGM, there are no intra-metafile controls to alter the navigation graph. In a scenario where such capability is desired, the private-namespace extension feature of the XCF can be used to introduce further intelligence associated with the contents of the metafile proper.

E.3 Required text content

Web Content Accessibility Guidelines such as [WCAG10] require that essential information be available in text form. Some key examples are:

- text alternatives for all text content, such as the 'alt' attribute on an 'img' element in HTML and XHTML.
- labels for form controls such as the 'input' element type in HTML.
- the text content of hyperlinks should provide a standalone orientation to where you are going if you follow the link.

WebCGM contains attributes that associate text with things in the depicted scene, and can be used to meet these requirements.

- The 'content' ensures that when there is text in a bitmap, the text content is available as computer text.
- The 'screentip' attribute is allowed on all objects (APS) Its behavior and usage parallel that of the 'title' attribute in HTML.
- The 'layerdesc' attribute is allowed on 'layer' APS types. This should be used roughly as the 'longdesc' attribute in HTML
- On hyperlinks ('linkuri' APS attribute), the attribute components "Link Title" (in metafile instances) or 'desc' (in XCF instances) provide a textual identification of where the link goes. Creators of WebCGM instances should ensure that the content available through these features provides a good orientation to the link destination.

This essential content, if not always presented to the user, must be considered 'conditional content' in the sense of [UAAG10], Checkpoint 2.3. That checkpoint gives some latitude to the viewer as to whether to present these attributes globally through view-mode controls or locally in response to focus and inspect actions of the user.

WebCGM viewers should also make this textual information available to assistive technology through the accessibility API appropriate to the programming platform, following [UAAG10], Guideline 6.

Note. It might be thought that the 'name' APS attribute could or should be used in a manner like 'alt' on 'img' in HTML. This is not the design intent of the 'name' attribute. It has well-defined category, not instance, semantics and associated categorical navigation behavior.

E.4 Styling

WebCGM versions 2.0 and higher support transient, display-time control of a number of display properties of objects. These can be manipulated via the WebCGM DOM, and are also available via the XCF mechanism. Although WebCGM does not support conventional styling, an (external) XCF can function similarly to an external style sheet. For improved accessibility, WebCGM viewers should provide some user control of rendering. Use of these mechanisms partially supports the recommendations of [WCAG10], Checkpoint 3.3 and [UAAG10], Guideline 4.

E.5 Visibility and navigation

By default, WebCGM viewers allow the user to navigate to and interact with only enabled elements (i.e. element whose 'visibility' attribute is 'on'). Objects which are not visible do not display tooltips (the 'screentip' APS attribute), may not be highlighted without making them visible, and may not be navigated to via the picture behaviors (whether in picture fragments or DOM src parameter).

In addition, WebCGM viewers can offer a mode where, at user option, the 'visibility' attribute is ignored, for accessibility or debugging support. It meets a requirement of [UAAG10], CheckPoint 9.3.

F. Regex search

This section is informative (non-normative).

The advanced example in this section shows how the functionality of the WebCGMAppStructure interface can be used to build useful convenience functions, in this case a function to search a WebCGM instance for all APS that contain an APS Attribute of a given type, whose value matches a given regular expression (regex).

EXAMPLE:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
  <title>WebCGM 2.0 regex example</title>
  <script type="text/ecmascript">
    var cgm;
    var pic;
    var mydiv;
    var o;
```

```
    var nl;
    var res;

    function loadCGM(fname) {
      try {
        var obj = document.getElementById('image');
        obj.innerHTML = "<object id='cgm' type='image/cgm;Version=4;ProfileId=WebCGM'
width='400' height='300' src='"+fname+"'>";
        document.getElementById('cgm').getWebCGMDocument().src = fname;
        cgm = document.getElementById('cgm').getWebCGMDocument();
        pic = cgm.firstPicture;
        mydiv = '<h2>Looking at file '+fname+'</h2>\n';
        mydiv += '<p>Enter an attribute and regular expression then click the Submit button.</p>
\n';
        mydiv += '<form name="myform">\n';
          mydiv += '<select name="attr">\n';
            mydiv += '<option value="content">content</option>\n';
            mydiv += '<option value="interactivity">interactivity</option>\n';
            mydiv += '<option value="layerdesc">layerdesc</option>\n';
            mydiv += '<option value="layername">layername</option>\n';
            mydiv += '<option value="linkuri">linkuri</option>\n';
            mydiv += '<option value="name">name</option>\n';
            mydiv += '<option value="region">region</option>\n';
            mydiv += '<option value="screentip">screentip</option>\n';
            mydiv += '<option value="viewcontext">viewcontext</option>\n';
            mydiv += '<option value="visibility">visibility</option>\n';
            mydiv += '</select>\n';
          mydiv += '<input type="text" name="regex" size="20"></input>\n';
          mydiv += '<input type="button" value="Submit" onclick="processAttr(attr.value, regex.
value)"></input>\n';
        mydiv += '</form>\n';
        var t =  document.getElementById('attinput');
        t.innerHTML = mydiv;
      } catch (e) {
        alert("loadCGM error " + e);
      }
    }

    function processAttr(att, val) {
      try {
        o = '<p>Searching APS for attribute '+att + ' containing '+val+'</p>\n';
        // Create a list of nodes and print out the attribute
        // This part is not really needed it is mostly here as
        // a sanity check.
        nl = listNodes(pic.firstChild);
        o+='<p>List of all nodes in the DOM tree.  A "" implies normally means the attribute
was not specified.</p><ul>';
        for(i=0; i< nl.length; i++){
          o+='<li>'+nl[i].apsId+' ('+att+'="'+nl[i].getAppStructureAttr(att)+'")</li>';
        }
        o+='</ul>';

        // Now we just run through the list and remove nodes that do not match
        res = new Array();
        for(i=0; i<nl.length;i++) {
          attVal = nl[i].getAppStructureAttr(att);
          if(attVal.match(val)) {
            res.push(nl[i]);
          }
        }

        if( res.length < 1) {
          o += '<p>No nodes found matching that regular expression</p>\n';
        } else {
          o+= '<p>The following nodes match the regular expression</p><ul>';
          for(i=0; i< res.length; i++){
            o+='<li>'+res[i].apsId+' ('+att+'="'+res[i].getAppStructureAttr(att)+'")</li>';
          }
```

```
          o+='</ul>';
        }
        var t = document.getElementById('attout');
        t.innerHTML = o;
      } catch(e) {
        alert("processAttr error " + e);
      }
    }

// These are helper functions that I found on the web
// http://www.ibm.com/developerworks/xml/library/x-matters41.html
//
// They are needed because you need to walk the tree to
// create a list of all of the nodes in the tree.

// return next node in document order
function nextNode(node) {
    if (!node) return null;
    if (node.firstChild){
        return node.firstChild;
    } else {
        return nextWide(node);
    }
}
// helper function for nextNode()
function nextWide(node) {
    if (!node) return null;
    if (node.nextSibling) {
        return node.nextSibling;
    } else {
        return nextWide(node.parentNode);
    }
}

// return an WebCGMNodeList of all nodes, starting at startNode and
// continuing through the rest of the DOM tree
function listNodes(startNode) {
    var node = startNode;
    var list = new Array();
    while(node) {
        list[list.length] =node;
        node = nextNode(node);
    }
    return list;
}

  </script>
</head>

<body>
<h1>WebCGM 2.0 regex example</h1>
<table>
  <tr>
    <td id="image">Select a CGM image...</td>
  </tr>
</table>
<br>
<input size="60" type="file" name="cgmfile" onChange="loadCGM(this.value)"/>
<div id="attinput">
</div>
<div id="attout">
</div>
</body>
</html>
```

View this example as HTML-CGM (WebCGM-DOM-enabled browsers only.)

Back to top of Appendixes