INTERNATIONAL STANDARD ISO/IEC 10180:1995
TECHNICAL CORRIGENDUM 1

Published 2001-10-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Processing languages — Standard Page Description Language (SPDL)

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Langages de traitement — Langage de description de page normalisée (SPDL)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to International Standard ISO/IEC 10180:1995 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 34, *Document description and processing languages*.

Introduction

Change "font object format" in list h) into "Indexed Font Representation Format".

3 Definitions

3.28 Dictionary

Change "name/value pairs" into "key/value pairs".

3.99 Printable String type

Change "ISO 8824" into "ISO/IEC 8824".

ICS 35.240.30 **Ref. No. ISO/IEC 10180:1995/Cor.1:2001(E)**

© ISO/IEC 2001 – All rights reserved

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

3.109 Simple Glyph Name type

Change "A type whose values are a subset of the values of the Name type" into "The type including subset of value of Name type. The set of values of the Simple Glyph Name type is the set of Names of length less than or equal to 100 characters which begin with a character other than the period (".")".

4 SPDL General Architecture

4.3.5 Document Presentation

Change "See figure 3)" in the 1st para., the last line into "See figure 3".

6 Types

6.10 Octet String type

Change "OctetString type" in NOTE 21 into "Octet String type".

7 Document Structure

7.7.7 Color Space Families Used

Change "used or" into "specified in".

7.7.7.2 HINT VALUE

Change "COLOR SPACE FAMILY" into "COLOR SPACE FAMILY IDENTIFIER".

8 External Structure Elements

Change "the within" in the 3rd para., line 3 into "within".

8.1 INCLUDABLE STRUCTURE

Change "FONTTYPE1 SPECIFICATION" into "FONTTYPE1 FONT SPECIFICATION".

9 Resource Definitions

9.1.5 Resource Declaration

Change "RESOURCE DEFINITION" in the last para., line 3 into "RESOURCE DECLARATION".

9.4.2.2 Name Binding

Change "RESOURCE DEFINITION" in line 3 into "RESOURCE DECLARATION".

10 Fonts

10.9.1 REMAP

Change "GLYPH INDEX MAP TRANSLATION TABLE" into "GLYPH INDEX TRANSLATION TABLE".

10.9.3 REMAPPED FONT SPECIFICATION Processing

Change "GLYPH INDEX MAP TRANSLATION TABLE" in item c into "GLYPH INDEX TRANSLATION TABLE".

© ISO/IEC 2001 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.10.3.2 INDEX VALUE

Change "the MAP SIZE structure element" in line 3 into "the preceding FONT INDEX MAP SIZE structure element".

10.12.2 GLYPH IDENTIFIER LIST

Change "the most immediately superior MAP SIZE structure element" in the 2nd para. into "the preceding GLYPH INDEX MAP SIZE structure element".

11 Document Production Instructions

11.5.1.4 NUMERIC DIMENSION

Change "NUMERIC DIMENSION VALUE" into "NUMERIC DIMENSION".

11.5.3.2 NAMED HEAD LOCATIONS

In the mathematical formulae, replace "σ" with "≤".

11.5.3.3 NUMERIC HEAD LOCATIONS

In the mathematical formulae, replace "σ" with "≤".

12 Medium Document Production Instructions

12.5.1 MEDIUM SIZE

Change "medium-size property" into "medium-size medium property".

13 Presentation Document Production Instructions

13.3.2 Supplementary Page Select Document Production Instruction Processing

Change "page-select" in the 2nd para., the last line into "supplementary-page-select".

13.7 Y-Image-Shift Document Production Instruction

Change "y-axis" in line 2 into "x-axis".

14 Finishing Document Production Instruction

14.2.7 Head Locations

Change "x-axis" in the 2nd para., line 3 into "y-axis".

Change "y-axis" in the 2nd para., line 4 into "x-axis".

14.4 FINISHING PROCESS SPECIFICATION

Change "the specification" in the 2nd para. into "the specification of".

14.7.5 Trimming Finishing Process

Change "the Reference Edge" in the 2nd para., line 3 into "the Jog Edge".

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Add "JOG EDGE" with arrow which shows the location of the JOG EDGE onto the underside of the rectangle in the Figure 7 — Trimming Finishing Process.

14.8.5 Die Cut Finishing Process

Change "the die cut itself" in line 4 into "the die itself".

14.12.11 Inserting Finishing Process

Change "the Medium Document Production" in the 2nd para., line 1 into "the Medium Document Production Instruction".

15 Management Document Production Instructions

15.8.7 Output Specification Document Production Instruction Reconciliation

Change "in Supplementary DPI" in the 2nd para., line 1 into "occurs in Supplementary DPI".

16 Document Structure and Content Processing

16.1 BLOCK

Change "a PAGES structure element" into "a PAGE SET structure element".

16.2.2.4 Initial CurrentTransformation

Change the reference "19.16" into "19.17".

17 Document Content Processing Model

Change "the content notation specifier in the Content value" in the 2nd para., line 4 into "the content notation identifier in the Current Context of Interpretation".

17.1.2 Parser

Change "the content notation specifier in the Content value" in line 5 into "the content notation identifier in the Current Context of Interpretation".

17.1.3 Sequence of tokens

Change "Vector object" in the 3rd para., line 2 and 3 into "Vector".

Change "Procedure object" into "Procedure".

17.2.3 Clipping Region

Change "assured reproduction area" in NOTE 81, line 1 and 4 into "imageable region (see 3.54)".

17.3.1 Reference Coordinate System

Change "16.8" in the last line into "19.19".

17.4.2 Objects

Delete the last sentence: "When the type of a object is implied by the content and there are a number of such objects in close proximity, the content literal may stand alone, without its associated type, as in the last example above."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
© ISO/IEC 2001 – All rights reserved
Not for Resale

17.4.8 Operand Stack

Change "the stack" in example, line 1 into "the operand stack".

18 Content Data Types

18.1.1 Any

Change "super type" into "super-type".

18.1.4 Dictionary Key

Change "18.2.4" in NOTE 84, line 1 into "18.2.1".

21 Stack and Composite Object Operators

21.1.13 Index

Change the number of results "n+1" into "n+2".

21.3.2 CheckIfReadable

Change "ReadOnly.," in the last para., line 1 into "ReadOnly,".

22 Flow of Control, Procedure, and State Operators

22.11 RestoreGraphicsState

Change "(see clause 12)" in the 2nd line into "(see 19)".

25 Base Indexed Font Specification Dictionaries

Change subclause titles in clause 25 as shown below:

- Change "25.1 FontType 3 Indexed Font Specification Dictionaries" into "25.1 Required key/value pairs for FontType 3 Indexed Font Specification Dictionaries".
- Change "25.2 FontType 3" into "25.2 Optional key/value pairs for FontType 3 Indexed Font Specification Dictionaries".
- Change "25.3 FontType 1 Indexed Font Specification Dictionaries" into "25.3 Required key/value pairs for FontType 1 Indexed Font Specification Dictionaries".
- Change "25.4 FontType 1 Indexed Font Specification Dictionary" into "25.4 Optional key/value pairs for FontType 1 Indexed Font Specification Dictionaries".
- Change "25.5 FontType 1 Indexed Font Specification Dictionaries" into "25.4.6 Other entries in FontType 1 Indexed Font Specification Dictionaries".

27 Font and Character Text Operators

27.1.2 FindFont

Change "SATISFACTION REQUIREMENT" into "SATISFACTION CRITERION".

29 Geometric Graphics Operators

29.4.2 RectangleStroke

Change the term "CTM" into "CurrentTransformation Graphics State Variable".

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

29.5.10 SetStrokeEnd

Change the column name "stroke ends" in Table 34 into "Stroke End Features".

29.5.11 SetStrokeJoin

Change the column name "mitre join" in Table 35 into "Stroke Join Features".

35.3.1.4 TransformPQR

Replace the last equation with

$$\begin{pmatrix} X_P & Y_P & Z_P \\ X_Q & Y_Q & Z_Q \\ X_R & Y_R & Z_R \end{pmatrix} = \begin{pmatrix} P_X & Q_X & R_X \\ P_Y & Q_Y & R_Y \\ P_Z & Q_Z & R_Z \end{pmatrix}^{-1}$$

36.6 Interpreter Errors

Change "Most" in the 1st para., line 1 into "most".

36.6 Interpreter Errors

Change "occurl" in the 1st para., line 3 into "occur".

37 Clear Text Representation and Interchange Format

37.1 Clear Text Structure Representation and Interchange Format

Change "are not allowed in content are not allowed" in the 2nd para., line 4 into "are not allowed".

37.1.2 SPDL Document Type Definition

Change "be defining" in NOTE 231, line 1 into "by defining".

38 Binary Representation and Interchange Format

38.1 Binary Structure Representation and Interchange Format

Change "specified in 38.2 through 38.2.6" in the 2nd para. into "specified in 38.1.1 through 38.1.1.6".

38.2 through 38.8

Renumber subclauses 38.2 through 38.8 as 38.1.1 through 38.1.7 respectively.

38.9 Binary Content Representation and Interchange Format

Renumber subclause 38.9 as 38.2.

38.9.3.4 Fixed Point Real Number value field

Change "value of of" in line 2 into "value of".

38.9.3.7 Procedure value field

Change "the the value" in the line 2 into "the value".

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2001 – All rights reserved

39 Conformance

39.5.2 Resource Definitions which are Subordinate to a DOCUMENT

Change "no more than than" in list item 1, line 2 and 3 into "no more than".

39.5.4 Color Space Families

Change "which which" in para. after Table 48, line 1 into "which".

39.8.1 Ability to construct and save objects

Change "name/value pairs" in item 4 into "key/value pairs".

39.10 Imaging

Change "in terms or" in line 1 into "in terms of".

Annex B

Remove the clause number and title "**B.1 Object Identifiers and SGML Formal Public Identifiers defined by this International Standard**". In all subsequent subclauses, change "B.1" into "B".

B.1.2 ASN.1 Object Identifier Values

Change "specified in 38.1.8" into "specified in 38.1.1.8".

Annex E

Table E.1

Change the glyph ID "afii:64" in (index 64) into "afii64".

Annex F

Remove the clause number and title "**F.1 ASN.1 Object Identifiers and Public Identifiers for Document Production Instructions defined by ISO/IEC 10175**". In all subsequent subclauses, change "F.1" into "F".

Table F.13

Change the object name "dpi::bindingtype::perfect" into "dpi::binding::type::perfect".

F.1.2

Change the "ISO 9070" and "ISO/IEC 8879" into "ISO/IEC 9070" and "ISO 8879" respectively.

© ISO/IEC 2001 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642444 180

INTERNATIONAL STANDARD

ISO/IEC 10180

First edition
1995-12-15

Information technology — Processing languages — Standard Page Description Language (SPDL)

Technologies de l'information — Langages de traitement — Langage de description de page normalisée (SPDL)



Reference number
ISO/IEC 10180:1995(E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Contents

Page

1 Scope 1

1.1 Scope 1

1.2 Field of Application 1

1.3 Relationship to Other Standards 2

2 Normative References 2

3 Definitions 3

4 SPDL General Architecture 9

4.1 Document Processing Model 9

4.1.1 Creation and Editing Process 9

4.1.2 Revisable Form Document 10

4.1.3 Composition and Layout Process 10

4.1.4 Final Form Document 11

4.1.5 Presentation Process 11

4.2 SPDL Document 11

4.2.1 Sources of SPDL Documents 11

4.2.2 SPDL Document Presentation 11

4.2.3 Uses of SPDL Documents 12

4.3 SPDL Document Architecture 12

4.3.1 Document Structure 12

4.3.2 Document Content 13

4.3.3 External structure elements 13

4.3.4 Resources 14

4.3.5 Document Presentation 14

© ISO/IEC 1995
All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.
ISO/IEC Copyright Office • Case Postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642446 T53

4.3.6 Representation and Interchange Format 14
4.4 Relationship to Print Services 15
4.4.1 Document Production Instructions 15
4.4.2 Relationship to ISO/IEC 10175 16
5 SPDL Structure 17
5.1 Hierarchical structure 17
5.1.1 Structure elements 17
5.1.2 Composite structure elements 17
5.1.3 Base structure elements 17
5.1.4 Subordinates and immediate subordinates 17
5.1.5 Superiors and immediate superiors 18
5.1.6 Most immediately superior structure elements 18
5.1.7 Peers 18
5.1.8 Highest structure level 18
5.1.9 Document Structure Elements 18
5.1.10 Scope 18
5.2 Sequential Order 19
5.2.1 Sequential Order and Hierarchical Structure 19
5.2.2 Example 19
5.2.3 Presentation Order 19
5.2.4 Ordinal Page Number 19
5.2.5 Pages and Imposition 19
5.3 Identification of Information Objects 20
5.3.1 Public Object Identifiers 20
5.3.2 Assignment of Public Object Identifiers 21
5.3.3 Notation 21
5.3.4 Environment Names 21
5.4 Representation and Interchange Format 22
5.5 Instances of the Standard Page Description Language 23

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642447 99T

5.5.1 Binary Structure Representation and Interchange Format 23

5.5.2 Clear Text Structure Representation and Interchange Format 23

5.5.3 TOP LEVEL STRUCTURE 23

5.5.4 Comments 24

6 Types 24

6.1 Boolean type 24

6.2 Enumerated types 24

6.3 Integer type 25

6.4 Cardinal type 25

6.5 Positive Integer Type 25

6.6 Real type 25

6.7 Number type 25

6.8 Non-Negative Number type 25

6.9 Printable String type 26

6.10 Octet String type 26

6.11 Name type 26

6.12 Object Identifier type 26

6.13 Public Identifier type 26

6.14 Public Object Identifier type 26

6.15 Environment Name type 27

6.16 Environment Identifier type 27

6.16.1 Structured Glyph Name type 27

6.16.2 ISO10036 Glyph Name type 27

6.16.3 Simple Glyph Name type 27

6.16.4 AFII Glyph Name type 28

6.16.5 Glyph Identifier type 28

7 Document Structure 28

7.1 DOCUMENT 28

7.2 PAGESET 29

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642448 826

7.2.1 PAGESET Processing 29
7.3 PAGE 29
7.4 PICTURE 30
7.4.1 CONTENT REPRESENTATION IDENTIFIER 30
7.4.2 PICTURE BODY 30
7.4.3 NON-SPDL PICTURE BODY 31
7.4.4 SPDL PICTURE Processing 31
7.4.5 NON-SPDL PICTURE Processing 31
7.5 PROLOGUE 32
7.6 INFORMATIVE DECLARATION 32
7.6.1 HINT 32
7.6.2 HINT NAME 33
7.6.3 HINT VALUE 33
7.6.4 INFORMATIVE DECLARATION Processing 33
7.7 Hints 33
7.7.1 Bounding Box 33
7.7.2 Number of Pages 34
7.7.3 Orientation 34
7.7.4 Font References 35
7.7.5 External References 35
7.7.6 Resources Used 35
7.7.7 Color Space Families Used 36
7.7.8 Media Used 36
7.7.9 Colorant Sets Used 36
7.8 NON-SPDL OPERATION 37
7.8.1 OPERATION NAME 37
7.8.2 OPERATION VALUE 37
7.8.3 NON-SPDL OPERATION Processing 37
7.9 CONTEXT DECLARATION 37

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642449 762

7.9.1 CONTEXT DECLARATION Processing ... 38
7.10 CONTEXT ADDITION ... 38
7.10.1 CONTEXT ADDITION Processing ... 38
7.11 SETUP PROCEDURE ... 39
7.11.1 SETUP PROCEDURE Processing ... 39
8 External Structure Elements ... 39
8.1 INCLUDABLE STRUCTURE ... 40
8.2 EXTERNAL DECLARATION ... 41
8.2.1 STRUCTURE ELEMENT CLASS IDENTIFIER ... 41
8.2.2 STRUCTURE ELEMENT IDENTIFIER ... 42
8.2.3 EXTERNAL DECLARATION Processing ... 42
8.3 LOCATION IDENTIFIER ... 43
8.3.1 PUBLIC LOCATION IDENTIFIER ... 43
8.3.2 LOCAL LOCATION IDENTIFIER ... 44
8.3.3 ENCODING DEPENDENT LOCATION IDENTIFIER ... 44
8.4 Referenced Structure Element Identification ... 45
8.5 EXTERNAL REFERENCE ... 46
8.5.1 EXTERNAL REFERENCE Processing ... 46
9 Resource Definitions ... 46
9.1 General ... 47
9.1.1 Resource Definition ... 47
9.1.2 Resource Identification ... 47
9.1.3 Resource Undefinition ... 47
9.1.4 Resource Reference Objects ... 48
9.1.5 Resource Declaration ... 48
9.2 ENVIRONMENT RESOURCE ... 48
9.2.1 RESOURCE UNDEFINITION ... 49
9.2.2 ENVIRONMENT RESOURCE Processing ... 49
9.3 RESOURCE DEFINITION ... 50

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642450 484

9.3.1 RESOURCE CLASS IDENTIFIER 50
9.3.2 ENVIRONMENT RESOURCE IDENTIFIER 51
9.3.3 RESOURCE SPECIFICATION 51
9.3.4 RESOURCE DEFINITION Processing 51
9.4 RESOURCE DECLARATION 52
9.4.1 INTERNAL RESOURCE IDENTIFIER 52
9.4.2 RESOURCE DECLARATION Processing 52
9.5 Context Dictionary 53
9.5.1 DICTIONARY SPECIFICATION 53
9.5.2 DICTIONARY SPECIFICATION Processing 53
9.6 Color Space 54
9.6.1 Color Space Family Identifiers 54
9.6.2 COLOR SPACE SPECIFICATION 55
9.6.3 COLOR SPACE FAMILY IDENTIFIER 55
9.6.4 PRIMARY SET SPECIFICATION 55
9.6.5 PRIMARY SET IDENTIFIER 56
9.6.6 PRIMARIES LIST 56
9.6.7 PRIMARY COLOR IDENTIFIER 56
9.6.8 COLOR SPACE SPECIFICATION Processing 56
9.6.9 Standard Color Space Resources 56
9.7 Data Source 57
9.7.1 DATA SOURCE SPECIFICATION 57
9.7.2 DATA BLOCK 57
9.7.3 DATA SOURCE SPECIFICATION Processing 57
9.7.4 Standard Data Source Resources 58
9.8 Filter 58
9.8.1 Filter RESOURCE SPECIFICATION 59
9.8.2 Standard Filters 59
9.9 Pattern 59

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642451 310

9.9.1 PATTERN SPECIFICATION **59**

9.9.2 PATTERN SPECIFICATION Processing **59**

9.10 Form **60**

9.10.1 FORM SPECIFICATION **60**

9.10.2 FORM SPECIFICATION Processing **60**

10 Fonts **60**

10.1 General **61**

10.1.1 Font Resources **61**

10.1.2 Glyph Index Maps **61**

10.1.3 Indexed Fonts **61**

10.1.4 Other Indexed Fonts **61**

10.2 Representation of Glyph Identifiers **62**

10.2.1 Conversion of ISO10036 Glyph Names **62**

10.2.2 Conversion of Other Structured Glyph Names **62**

10.2.3 Conversion of Simple Glyph Names **62**

10.3 Font References **63**

10.3.1 Font Resource Identifiers **63**

10.3.2 Font Properties **63**

10.3.3 Unsatisfied Font References **63**

10.4 FONT SPECIFICATION **64**

10.5 FONT IDENTIFIER FONT SPECIFICATION **64**

10.6 FONT REFERENCE FONT SPECIFICATION **64**

10.6.1 GLYPH INDEX MAP IDENTIFIER **64**

10.6.2 FONT REFERENCE FONT SPECIFICATION Processing **65**

10.7 FONT REFERENCE **65**

10.7.1 FONT RESOURCE IDENTIFIER **65**

10.7.2 REQUIRED PROPERTIES **66**

10.7.3 ADVISORY PROPERTIES **66**

10.7.4 MATCH RULES **66**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642452 257

10.7.5 SATISFACTION CRITERION **66**

10.7.6 FONT REFERENCE Processing **67**

10.7.7 Standard FONT RESOURCE IDENTIFIER Values **68**

10.8 FONTTYPE1 FONT SPECIFICATION **68**

10.9 REMAPPED FONT SPECIFICATION **68**

10.9.1 REMAP **69**

10.9.2 GLYPH INDEX TRANSLATION TABLE **69**

10.9.3 REMAPPED FONT SPECIFICATION Processing **69**

10.9.4 Errors and Warnings **70**

10.10 COMPOSITE FONT SPECIFICATION **70**

10.10.1 FMAPTYPE **70**

10.10.2 FMAPTYPE PARAMETERS LIST **70**

10.10.3 FONT INDEX MAP **71**

10.10.4 FONT LIST **72**

10.10.5 COMPOSITE FONT SPECIFICATION Processing **72**

10.11 USER FONT SPECIFICATION **73**

10.12 GLYPH INDEX MAP SPECIFICATION **73**

10.12.1 GLYPH INDEX MAP SIZE **73**

10.12.2 GLYPH IDENTIFIER LIST **73**

10.12.3 GLYPH INDEX MAP SPECIFICATION Processing **74**

10.12.4 Standard Glyph Index Maps **74**

11 Document Production Instructions **74**

11.1 General **74**

11.2 Sources of Document Production Instructions **75**

11.2.1 Document Structure **75**

11.2.2 Supplementary DPI **75**

11.2.3 Additional DPI **75**

11.2.4 Reconciliation of Document Production Instructions **76**

11.3 Classes of Document Production Instructions **76**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642453 193

11.4 Fallback 77

11.5 Useful Structure elements 77

11.5.1 DIMENSION 77

11.5.2 XYDIMENSIONS 78

11.5.3 HEAD LOCATIONS 79

11.5.4 EDGE 80

11.6 DPI DECLARATION 80

11.7 Additional Document Production Instructions 81

11.7.1 ADDITIONAL DPI 81

11.7.2 DPI NAME 81

11.7.3 DPI VALUE 82

11.7.4 ADDITIONAL DPI Semantics 82

12 Medium Document Production Instructions 82

12.1 Medium Declaration 83

12.1.1 Medium Specification 83

12.1.2 Selection of Physical Media 83

12.1.3 Medium-list presentation parameter 84

12.1.4 Default medium 84

12.2 Association of Media with Ordinal Page Numbers 84

12.2.1 Medium Selection by Medium Select 85

12.2.2 Medium Selection by Current Medium 85

12.3 Medium Document Production Instruction 85

12.3.1 MEDIUM DPI 85

12.3.2 MEDIUM Document Production Instruction Processing 86

12.3.3 Medium Document Production Instruction Reconciliation 86

12.3.4 Medium Document Production Instruction Fallback 87

12.4 MEDIUM SPECIFICATION 87

12.4.1 MEDIUM NAME 87

12.4.2 MEDIUM MESSAGE 87

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642454 02T

12.4.3 MEDIUM SPECIFICATION Processing 87

12.5 MEDIUM PROPERTIES 88

12.5.1 MEDIUM SIZE 88

12.5.2 MEDIUM COLOR 89

12.5.3 MEDIUM WEIGHT 89

12.5.4 MEDIUM TYPE 89

12.5.5 MEDIUM TOOTH 90

12.5.6 MEDIUM GRAIN 90

12.5.7 MEDIUM PRE-PUNCH 90

12.5.8 MEDIUM MULTI-PART 92

12.5.9 MEDIUM FINISHED EDGES 93

12.5.10 MEDIUM LABELS 93

12.5.11 ADDITIONAL PROPERTY 94

12.6 Medium Select Document Production Instruction 95

12.6.1 MEDIUM SELECT DPI 95

12.6.2 Medium Select Document Production Instruction Processing 96

12.6.3 Medium Select Document Production Instruction Reconciliation 96

12.6.4 Medium Select Document Production Instruction Fallback 96

12.7 Current Medium Document Production Instruction 96

12.7.1 CURRENT MEDIUM DPI 96

12.7.2 Current Medium Document Production Instruction Processing 96

12.7.3 Current Medium Document Production Instruction Reconciliation 96

12.7.4 Current Medium Document Production Instruction Fallback 97

13 Presentation Document Production Instructions 97

13.1 Copies Document Production Instruction 97

13.1.1 COPIES DPI 97

13.1.2 Copies Document Production Instruction Processing 97

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642455 T66

13.1.3 Copies Document Production Instruction Reconciliation 98

13.1.4 Copies Document Production Instruction Fallback 98

13.2 Page Select Document Production Instruction 98

13.2.1 PAGE SELECT DPI 98

13.2.2 Page Select Document Production Instruction Processing 99

13.2.3 Page Select Document Production Instruction Reconciliation 99

13.2.4 Page Select Document Production Instruction Fallback 100

13.3 Supplementary Page Select Document Production Instruction 100

13.3.1 SUPPLEMENTARY PAGE SELECT DPI 100

13.3.2 Supplementary Page Select Document Production Instruction Processing 100

13.3.3 Supplementary Page Select Document Production Instruction Reconciliation 101

13.3.4 Supplementary Page Select Document Production Instruction Fallback 101

13.4 Sides Document Production Instruction 101

13.4.1 SIDES DPI 101

13.4.2 Sides Document Production Instruction Processing 101

13.4.3 Sides Document Production Instruction Reconciliation 101

13.4.4 Sides Document Production Instruction Fallback 101

13.5 Plex Document Production Instruction 101

13.5.1 PLEX DPI 102

13.5.2 Plex Document Production Instruction Processing 102

13.5.3 Plex Document Production Instruction Reconciliation 102

13.5.4 Plex Document Production Instruction Fallback 102

13.6 X-Image-Shift Document Production Instruction 102

13.6.1 X-IMAGE SHIFT DPI 102

13.6.2 X-Image-Shift Document Production Instruction Processing 102

13.6.3 X-Image-Shift Document Production Instruction Reconciliation 102

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642456 9T2

13.6.4 X-Image-Shift Document Production Instruction Fallback 103
13.7 Y-Image-Shift Document Production Instruction 103
13.7.1 Y-IMAGE SHIFT DPI 103
13.7.2 Y-Image-Shift Document Production Instruction Processing 103
13.7.3 Y-Image-Shift Document Production Instruction Reconciliation 103
13.7.4 Y-Image-Shift Document Production Instruction Fallback 103
13.8 Current Side Document Production Instruction 103
13.8.1 CURRENT SIDE DPI 103
13.8.2 Current Side Document Production Instruction Processing 104
13.8.3 Current Side Document Production Instruction Reconciliation 104
13.8.4 Current Side Document Production Instruction Fallback 104
14 Finishing Document Production Instruction 104
14.1 Finishing Model 104
14.2 Finishing Process Presentation Parameters 106
14.2.1 Inherited Finishing Process Presentation Parameters . 107
14.2.2 Other Finishing Process Parameters 107
14.2.3 Reference Size 107
14.2.4 Reference Edge 108
14.2.5 Jog Edge 108
14.2.6 Process Offset 108
14.2.7 Head Locations 108
14.3 Finishing Document Production Instruction 109
14.3.1 FINISHING DPI 109
14.3.2 Finishing Document Production Instruction Processing 110
14.3.3 Finishing Document Production Instruction Reconciliation 110
14.3.4 Finishing Document Production Instruction Fallback 110
14.4 FINISHING PROCESS SPECIFICATION 110
14.5 STITCHING SPECIFICATION 111

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642457 839

14.5.1 NAMED STITCHING 111
14.5.2 STITCHING PARAMETERS 112
14.5.3 STITCH TYPE 112
14.5.4 Stitching Finishing Process 112
14.6 BINDING SPECIFICATION 112
14.6.1 NAMED BINDING 113
14.6.2 BINDING PARAMETERS 113
14.6.3 BINDING TYPE 113
14.6.4 BINDING COLOR 113
14.6.5 Binding Finishing Process 114
14.7 TRIMMING SPECIFICATION 114
14.7.1 NAMED TRIMMING 114
14.7.2 TRIMMING PARAMETERS 114
14.7.3 TRIM OFFSET 114
14.7.4 TRIM DIMENSIONS 114
14.7.5 Trimming Finishing Process 115
14.8 DIE CUTTING SPECIFICATION 115
14.8.1 NAMED DIE CUTTING 115
14.8.2 DIE CUTTING PARAMETERS 115
14.8.3 DIE CUT POSITION 116
14.8.4 DIE CUT NAME 116
14.8.5 Die Cut Finishing Process 117
14.9 PUNCHING SPECIFICATION 117
14.9.1 NAMED PUNCHING 117
14.9.2 PUNCHING PARAMETERS 117
14.9.3 PUNCH DIAMETER 117
14.9.4 Punching Finishing Process 118
14.10 PERFORATING SPECIFICATION 118
14.10.1 NAMED PERFORATION 119

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642458 775

14.10.2 PERFORATION PARAMETERS 119
14.10.3 PERFORATION TYPE 119
14.10.4 Perforating Finishing Process 119
14.11 SLITTING SPECIFICATION 120
14.11.1 NAMED SLITTING 120
14.11.2 SLITTING PARAMETERS 120
14.11.3 Slitting Finishing Process 120
14.12 INSERT SPECIFICATION 120
14.12.1 NAMED INSERTING 121
14.12.2 INSERTING PARAMETERS 121
14.12.3 INSERT SHEET LIST 121
14.12.4 INSERT SHEET 121
14.12.5 INSERT IDENTIFIER 121
14.12.6 INSERT NAME 122
14.12.7 INSERT BIN 122
14.12.8 INSERT TOP SURFACE 122
14.12.9 INSERT AFTER 122
14.12.10 INSERT MESSAGE 122
14.12.11 Inserting Finishing Process 123
14.13 COVERS SPECIFICATION 123
14.13.1 NAMED COVERS 123
14.13.2 COVERS PARAMETERS 123
14.13.3 FRONT COVER 123
14.13.4 BACK COVER 124
14.13.5 COVER NAME 124
14.13.6 Covers Finishing Process 124
14.14 FOLDING SPECIFICATION 124
14.14.1 NAMED FOLDING 124
14.14.2 FOLDING PARAMETERS 124

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642459 601

14.14.3 Folding Finishing Process 125

14.15 OTHER FINISHING SPECIFICATION 125

14.15.1 FINISHING OP NAME 125

14.15.2 FINISHING OP PARAMETERS 125

14.15.3 NAMED FINISHING OP PARAMETERS 125

14.15.4 SPECIFIED FINISHING OP PARAMETERS 126

14.15.5 ADDITIONAL PARAMETERS 126

14.15.6 OTHER FINISHING SPECIFICATION Finishing Processes 126

15 Management Document Production Instructions 126

15.1 Colorant Set Document Production Instruction 127

15.1.1 COLORANT SET DPI 127

15.1.2 Colorant Set Document Production Instruction Processing 129

15.1.3 Colorant Set Document Production Instruction Reconciliation 129

15.1.4 Colorant Set Document Production Instruction Fallback 129

15.2 Document Start Message Document Production Instruction 129

15.2.1 DOCUMENT START MESSAGE DPI 129

15.2.2 Document Start Message Document Production Instruction Processing 129

15.2.3 Document Start Message Document Production Instruction Reconciliation 129

15.2.4 Document Start Message Document Production Instruction Fallback 129

15.3 Auxiliary Page Type Document Production Instruction .. 130

15.3.1 AUXILIARY PAGE TYPE DPI 130

15.3.2 START PAGE TYPE 130

15.3.3 SEPARATOR PAGE TYPE 131

15.3.4 END PAGE TYPE 131

15.3.5 Auxiliary Page Type Document Production Instruction Processing 131

15.3.6 Auxiliary Page Type Document Production Instruction Reconciliation 131

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642460 323

15.3.7 Auxiliary Page Type Document Production Instruction Fallback 132

15.4 Document Comment Document Production Instruction .. 132

15.4.1 DOCUMENT COMMENT DPI 132

15.4.2 Document Comment Document Production Instruction Processing 132

15.4.3 Document Comment Document Production Instruction Reconciliation 132

15.4.4 Document Comment Document Production Instruction Fallback 132

15.5 Timeout Document Production Instruction 132

15.5.1 TIMEOUT DPI 132

15.5.2 Timeout Document Production Instruction Processing 132

15.5.3 Timeout Document Production Instruction Reconciliation 133

15.5.4 Timeout Document Production Instruction Fallback 133

15.6 Document Abort Policy Document Production Instruction 133

15.6.1 ABORT POLICY DPI 133

15.6.2 Document Abort Policy Document Production Instruction Processing 133

15.6.3 Document Abort Policy Document Production Instruction Reconciliation 134

15.6.4 Document Abort Policy Document Production Instruction Fallback 134

15.7 Output Bin Document Production Instruction 134

15.7.1 OUTPUT BIN DPI 134

15.7.2 Output Bin Document Production Instruction Processing 135

15.7.3 Output Bin Document Production Instruction Reconciliation 135

15.7.4 Output Bin Document Production Instruction Fallback . 135

15.8 Output Specification Document Production Instruction .. 135

15.8.1 OUTPUT SPECIFICATION DPI 135

15.8.2 COLLATED 136

15.8.3 OFFSET 136

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642461 26T

15.8.4 BURST **136**

15.8.5 ADDITIONAL OUTPUT **137**

15.8.6 Output Specification Document Production Instruction Processing **137**

15.8.7 Output Specification Document Production Instruction Reconciliation **137**

15.8.8 Output Specification Document Production Instruction Fallback **138**

15.9 Document End Message Document Production Instruction **138**

15.9.1 DOCUMENT END MESSAGE DPI **138**

15.9.2 Document End Message Processing **138**

15.9.3 Document End Message Reconciliation **138**

15.9.4 Document End Message Fallback **138**

16 Document Structure and Content Processing **138**

16.1 BLOCK **139**

16.2 Block State **139**

16.2.1 Current Virtual Machine State **139**

16.2.2 Initial Block State of a TOP LEVEL STRUCTURE **140**

16.2.3 Initial Block State of a Subordinate BLOCK **141**

16.3 Current Page Image **143**

16.4 Interface to the Content Processor **143**

16.4.1 Content value **143**

16.4.2 Context of Interpretation **144**

16.4.3 Current Page Image **145**

16.4.4 Content Processing **145**

16.4.5 Return **145**

16.5 TOKENSEQUENCE **146**

16.6 Association of Page Images with Instances of the Media **146**

16.6.1 Page Selection and Numbering **146**

16.6.2 Medium Declaration and Selection **147**

16.6.3 Current Side Presentation Parameter **147**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642462 1T6

16.6.4 Association of Page Images with Instances of the Media 148
16.6.5 One sided printing 148
16.6.6 Two sided printing 148
16.7 Placement of the Page Image on the Medium 149
16.7.1 Location and Orientation of the Reference Coordinate System on the Medium 149
16.7.2 Placement of the Page Image on the Imageable Surface of the Medium 150
16.7.3 Initial CurrentTransformation 151
17 Document Content Processing Model 152
17.1 Virtual Machine 153
17.1.1 Basic Concepts 154
17.1.2 Parser 156
17.1.3 Sequence of tokens 156
17.1.4 Interpreter 156
17.1.5 Operand Stack 157
17.1.6 Context Stack 157
17.1.7 Set of Referenced Objects 158
17.1.8 State Variables 158
17.1.9 Graphics State and the Graphics State Stack 159
17.1.10 Ordered set of SaveObjects 159
17.1.11 Imager 159
17.1.12 Current Page Image 160
17.1.13 Set of Declared Resources 160
17.1.14 Current Abort Policy 160
17.1.15 Initial *CurrentTransformation* 160
17.2 Imaging Model 160
17.2.1 Ink 161
17.2.2 Mask 161
17.2.3 Clipping Region 162

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642463 032

17.2.4 Page Image ... **162**

17.3 Coordinate Systems **163**

17.3.1 Reference Coordinate System **163**

17.3.2 User Coordinate Systems **163**

17.3.3 Initial *CurrentTransformation* **164**

17.4 Notation **164**

17.4.1 Literals **164**

17.4.2 Objects **164**

17.4.3 Object Types **165**

17.4.4 Abbreviated Forms **165**

17.4.5 Sequences of objects **166**

17.4.6 Operators **166**

17.4.7 State and Graphics State Variables **166**

17.4.8 Operand Stack **167**

17.4.9 Operator descriptions **167**

18 Content Data Types **168**

18.1 Value types **168**

18.1.1 Any **168**

18.1.2 Boolean **168**

18.1.3 Cardinal **168**

18.1.4 Dictionary Key **169**

18.1.5 Identifier **169**

18.1.6 Integer **169**

18.1.7 Mark **169**

18.1.8 Name **169**

18.1.9 Null **169**

18.1.10 Number **169**

18.1.11 Operator **170**

18.1.12 Real **170**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642464 T79

18.1.13 SaveObject ... 170
18.1.14 StreamObject ... 170
18.2 Composite object types ... 170
18.2.1 Dictionary ... 170
18.2.2 OctetString ... 171
18.2.3 Path ... 171
18.2.4 Vector ... 172
18.3 Object reference types ... 172
18.3.1 Indexed Font ... 172
18.3.2 Glyph String ... 173
18.3.3 Procedure ... 173
18.3.4 Transformation ... 173
18.4 Attributes ... 174
18.4.1 Executability Attribute ... 174
18.4.2 Access Attribute ... 174
19 State Variables ... 175
19.1 Current Black Generation ... 175
19.2 Current Clipping Region ... 175
19.3 Current Color ... 175
19.4 Current Color Rendering ... 176
19.5 Current Color Space ... 176
19.6 Current Dash Pattern ... 176
19.7 Current Font ... 176
19.8 Current Halftone ... 176
19.9 Current Miter Limit ... 176
19.10 Current Path ... 177
19.11 Current Position ... 177
19.12 Current *OverPrint* ... 177
19.13 Current Stroke Adjust ... 177

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642465 905

19.14 Current Stroke End **177**

19.15 Current Stroke Join **177**

19.16 Current Stroke Width **178**

19.17 Current Transformation **178**

19.18 Current UnderColor Removal **178**

19.19 Device Description Dictionary **178**

19.19.1 Process Color Class **178**

19.19.2 Current Resolution **179**

19.19.3 Current Medium Size **179**

19.19.4 Current Imageable Region **179**

20 Arithmetic and Logic Operators **180**

20.1 AbsoluteValue **180**

20.2 Add **180**

20.3 ArcTangent **181**

20.4 And **181**

20.5 Ceiling **182**

20.6 Cosine **182**

20.7 Divide **182**

20.8 Equal **182**

20.9 Exponentiate **183**

20.10 False **183**

20.11 Floor **183**

20.12 GreaterOrEqual **184**

20.13 GreaterThan **184**

20.14 IntegerDivide **184**

20.15 LessOrEqual **184**

20.16 LessThan **185**

20.17 Logarithm **185**

20.18 LogicalShift **185**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642466 841

20.19 Multiply 185

20.20 NaturalLogarithm 186

20.21 Negate 186

20.22 Not 186

20.23 NotEqual 186

20.24 Null 187

20.25 Or 187

20.26 Rand 187

20.27 RandSetState 187

20.28 Remainder 187

20.29 Round 188

20.30 Sine 188

20.31 SquareRoot 188

20.32 Subtract 188

20.33 True 189

20.34 Truncate 189

20.35 Xor 189

21 Stack and Composite Object Operators 190

21.1 Operators for Manipulating Values on the Operand Stack 190

21.1.1 ClearStack 190

21.1.2 ClearToMark 190

21.1.3 ConvertToExecutable 190

21.1.4 ConvertToIdentifier 190

21.1.5 ConvertToInteger 190

21.1.6 ConvertToReal 191

21.1.7 ConvertToString 191

21.1.8 Copy 192

21.1.9 Count 193

21.1.10 CountToMark 193

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642467 788

21.1.11 Dup **193**

21.1.12 Exchange **193**

21.1.13 Index **194**

21.1.14 Mark **194**

21.1.15 Pop **194**

21.1.16 Roll **195**

21.1.17 Type **196**

21.2 Operators for Manipulating Dictionaries, Vectors, and OctetStrings **196**

21.2.1 AnchorSearch **196**

21.2.2 Capacity **197**

21.2.3 ContextStack **197**

21.2.4 Define **197**

21.2.5 EntriesUsed **198**

21.2.6 Get **198**

21.2.7 GetCurrentDictionary **198**

21.2.8 GetInterval **199**

21.2.9 GetTest **199**

21.2.10 GetValue **199**

21.2.11 GetValueTest **199**

21.2.12 MakeDictionary **200**

21.2.13 MakeString **200**

21.2.14 MakeVector **200**

21.2.15 MakeandStoreDictionary **201**

21.2.16 MakeandStoreVector **201**

21.2.17 PopContextStack **202**

21.2.18 PushContextStack **202**

21.2.19 Put **202**

21.2.20 PutInterval **202**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642468 614

21.2.21 PutValue ... 203
21.2.22 Search ... 203
21.2.23 StoreVector ... 204
21.2.24 VectorLoad ... 204
21.3 Attribute and Resource Operators ... 204
21.3.1 CheckIfExecutable ... 204
21.3.2 CheckIfReadable ... 205
21.3.3 CheckIfWriteable ... 205
21.3.4 FindResource ... 205
21.3.5 QueryResource ... 207
21.3.6 MakeReadOnly ... 208
21.3.7 MakeExecuteOnly ... 208
22 Flow of Control, Procedure, and State Operators ... 208
22.1 Execute ... 208
22.2 Exit ... 209
22.3 For ... 209
22.4 ForAll ... 209
22.5 GetDeviceDescription ... 210
22.6 If ... 210
22.7 IfElse ... 210
22.8 Loop ... 211
22.9 Noop ... 211
22.10 Repeat ... 211
22.11 RestoreGraphicsState ... 211
22.12 RestoreGraphicsStateXCP ... 211
22.13 RestoreSavedGraphicsState ... 211
22.14 RestoreState ... 212
22.15 SaveGraphicsState ... 212
22.16 SaveState ... 212

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642469 550

23 Coordinate Transformation Operators **212**

23.1 ScaleT **213**

23.2 Scale **213**

23.3 TranslateT **213**

23.4 Translate **214**

23.5 RotateT **214**

23.6 Rotate **214**

23.7 ConcatT **214**

23.8 Concat **215**

23.9 SetTrans **215**

23.10 GetTrans **215**

24 Character Text and the Indexed Font Architecture **215**

24.1 Fonts **216**

24.1.1 Font Resources **216**

24.1.2 Font Size, scaling and rotation **216**

24.1.3 Bold and italic text **216**

24.1.4 Writing modes **217**

24.2 Glyphs, Glyph Mappings and Indexed Fonts **217**

24.2.1 Glyph identifiers **217**

24.2.2 Strings **217**

24.2.3 Indices **217**

24.2.4 Indexed Fonts **218**

24.2.5 Glyph Mappings **218**

24.3 Indexed Fonts **218**

24.3.1 Indexed Font Specification Dictionaries **218**

24.3.2 Indexed Fonts **221**

24.4 Indexed Font Dictionaries **222**

25 Base Indexed Font Specification Dictionaries **223**

25.1 FontType 3 Indexed Font Specification Dictionaries **223**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642470 272

25.1.1 FontMatrix ... 223

25.1.2 FontBBox ... 223

25.1.3 Encoding ... 223

25.1.4 Metrics ... 223

25.1.5 ConstructGlyph ... 224

25.2 FontType 3 ... 224

25.2.1 WMode ... 224

25.2.2 OtherMetrics ... 224

25.3 FontType 1 Indexed Font Specification Dictionaries ... 225

25.3.1 Key/Value pairs with same semantics as in FontType 3 225

25.3.2 PaintType ... 225

25.3.3 CharStrings ... 225

25.3.4 FontType 3 keys that are not present in FontType 1 ... 225

25.4 FontType 1 Indexed Font Specification Dictionary ... 226

25.4.1 Metrics ... 226

25.4.2 Metrics2 ... 227

25.4.3 OtherMetrics ... 227

25.4.4 CDevProc ... 227

25.4.5 WMode ... 228

25.5 FontType 1 Indexed Font Specification Dictionaries ... 228

25.6 Base Font Glyph Mapping ... 228

25.7 Standard GLYPH INDEX MAPs ... 229

25.7.1 Latin1 Publishing GLYPH INDEX MAP ... 229

25.7.2 Latin1 Publishing (A) GLYPH INDEX MAP ... 229

25.7.3 Algorithmic AFII GLYPH INDEX MAPs ... 229

26 Composite Indexed Font Specification Dictionaries ... 229

26.1 FontType 0 Indexed Font Specification Dictionaries ... 229

26.1.1 FontMatrix ... 229

26.1.2 Encoding ... 229

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642471 109

26.1.3 FDepVector **230**

26.1.4 FMapType **230**

26.2 Other Key/Value Pairs **230**

26.3 Optional Key/Value Pairs **230**

26.3.1 WMode **230**

26.4 Composite Font Glyph Mappings **230**

26.4.1 Composite Font Glyph Mapping Algorithms **231**

26.4.2 8/8 Mapping **232**

26.4.3 1/7 Mapping **233**

26.4.4 9/7 Mapping **233**

26.4.5 Interval Mapping **233**

26.4.6 Escape Mapping **234**

26.4.7 Double Escape Mapping **234**

26.4.8 ShiftOut/ShiftIn Mapping **235**

26.5 Standard FONT INDEX MAPs **235**

26.5.1 Algorithmic FONT INDEX MAPs **235**

27 Font and Character Text Operators **236**

27.1 Font Accessing and Manipulation Operators **236**

27.1.1 DefineFont **236**

27.1.2 FindFont **236**

27.1.3 GetRootFont **236**

27.1.4 GetSelectedFont **237**

27.1.5 OpenFont **237**

27.1.6 PutWMode **237**

27.1.7 ScaleFont **237**

27.1.8 TransformFont **238**

27.2 Graphics State Manipulation Operators **238**

27.2.1 GetPosition **238**

27.2.2 SetFont **238**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642472 045

27.2.3 SetPosition 239
27.2.4 SetPositionRelative 239
27.3 Character Text Operators 239
27.3.1 ShowGlyph 239
27.3.2 ShowString 241
27.3.3 ShowStringEscapedX 241
27.3.4 ShowStringEscapedY 242
27.3.5 ShowStringEscapedXY 242
27.3.6 StringWidth 243
28 Raster Graphics Operators 244
28.1 Model for Raster Graphics 244
28.1.1 Sampled Raster Graphics 245
28.1.2 Bitmap Raster Graphics 246
28.1.3 Raster Graphics Image Data 246
28.2 Image Dictionaries 246
28.2.1 Image Dictionary Contents and Semantics 246
28.2.2 Data Sources 248
28.2.3 In-Line Data 249
28.2.4 Decode Vector 250
28.2.5 Interpolation 250
28.3 Operators 250
28.3.1 ImageRasterElement 251
28.3.2 MaskBitMap 252
29 Geometric Graphics Operators 252
29.1 Model for Geometric Graphics 252
29.2 Path Construction and Manipulation Operators 253
29.2.1 AppendPath 253
29.2.2 ArcToClockwise 253
29.2.3 ArcToCounterClockwise 254

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642473 T81

29.2.4 ClosePathSegment **255**

29.2.5 CurveTo **255**

29.2.6 CurveToRelative **256**

29.2.7 GetPath **256**

29.2.8 GlyphToPath **256**

29.2.9 LineTo **257**

29.2.10 LineToRelative **257**

29.2.11 NewPath **257**

29.2.12 OutlineStroke **257**

29.2.13 SetPath **258**

29.3 Stroking and Filling Operators **258**

29.3.1 FillPath **258**

29.3.2 FillPathEvenOdd **258**

29.3.3 StrokePath **258**

29.4 Rectangular Fill and Stroke Operators **261**

29.4.1 RectangleFill **261**

29.4.2 RectangleStroke **261**

29.5 Graphics State Variable Operators **262**

29.5.1 GetDashPattern **262**

29.5.2 GetMiterLimit **263**

29.5.3 GetStrokeAdjust **263**

29.5.4 GetStrokeEnd **263**

29.5.5 GetStrokeJoin **263**

29.5.6 GetStrokeWidth **263**

29.5.7 SetDashPattern **263**

29.5.8 SetMiterLimit **264**

29.5.9 SetStrokeAdjust **264**

29.5.10 SetStrokeEnd **264**

29.5.11 SetStrokeJoin **264**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642474 918

29.5.12 SetStrokeWidth ... 265
30 Clipping Operators ... 265
30.1 Model for Clipping ... 265
30.2 Operators ... 266
30.2.1 ClipPath ... 266
30.2.2 ClipPathEvenOdd ... 266
30.2.3 RectangleClip ... 267
31 Filters ... 267
31.1 Model for Filters ... 268
31.1.1 Data Sources for Filters ... 268
31.1.2 End-of-Data ... 269
31.1.3 Filter Identifiers ... 269
31.2 Standard Filters ... 269
31.2.1 ASCIIHexDecode ... 269
31.2.2 ASCII85Decode ... 269
31.2.3 LZWDecode ... 270
31.2.4 RunLengthDecode ... 270
31.2.5 CCITTFaxDecode ... 270
31.2.6 NullDecode ... 271
31.3 Operators ... 272
31.3.1 Filter ... 272
32 Patterns ... 273
32.1 Creation and Use of Patterns ... 273
32.2 Pattern Dictionaries ... 273
32.2.1 PaintType ... 274
32.2.2 PaintProc ... 274
32.2.3 BBox ... 275
32.2.4 XStep ... 275
32.2.5 YStep ... 275

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0642475 854

32.2.6 Tiling **275**

32.2.7 TilingType **275**

32.2.8 Implementation **276**

32.3 Pattern Semantics **276**

32.3.1 Colored Patterns **276**

32.3.2 Mask Patterns **276**

32.4 Obtaining a Prototype Pattern Dictionary for Use in Content **277**

32.5 Operators **277**

32.5.1 MakePattern **277**

32.5.2 SetPatternColor **278**

33 Forms **278**

33.1 Model for Forms **278**

33.2 Form Dictionaries **278**

33.2.1 BBox **279**

33.2.2 Matrix **279**

33.2.3 PaintProc **279**

33.2.4 UniqueID **279**

33.2.5 Implementation **279**

33.3 Obtaining a Form Dictionary for Use in Content **280**

33.4 Imaging a Form **280**

33.5 ExecuteForm **280**

34 Color Space and Color Operators **282**

34.1 Model for Color Spaces and Colors **282**

34.2 Color Spaces and Color Space Families **282**

34.3 CIE and CIE-based Color Spaces **283**

34.3.1 CIELAB **283**

34.3.2 CIELUV **284**

34.3.3 CIEBasedABC **285**

34.3.4 CIEBasedA **289**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642476 790

34.4 Device Color Spaces **291**

34.4.1 DeviceRGB **292**

34.4.2 DeviceCMYK **292**

34.4.3 DeviceKX **293**

34.4.4 DeviceGray **293**

34.5 Special-Purpose Color Spaces **294**

34.5.1 Indexed **294**

34.5.2 NamedColor **295**

34.6 Operators **296**

34.6.1 SetColorSpace **296**

34.6.2 GetColorSpace **296**

34.6.3 SetColor **297**

34.6.4 GetColor **297**

34.6.5 SetOverPrint **297**

34.6.6 GetOverPrint **298**

35 Rendering and Rendering Control Operators **298**

35.1 Model for Rendering **298**

35.2 Conversions from CIE Color≻ to Device Color **299**

35.2.1 Transforming CIE Color to Device Color **299**

35.3 Color Rendering Dictionaries **300**

35.3.1 Color Rendering Type 1 Dictionary **301**

35.4 Conversions among Device Color Spaces **306**

35.4.1 Conversion between DeviceRGB and *DeviceGray* **307**

35.4.2 Conversion between DeviceCMYK and *DeviceGray* **307**

35.4.3 Conversion from DeviceRGB to DeviceCMYK **307**

35.4.4 Conversion from DeviceCMYK to DeviceRGB **309**

35.5 Transfer Functions **309**

35.5.1 Specifying Transfer Functions **309**

35.6 Halftone Functions **310**

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0642477 627

35.6.1 Model for Halftone Screens **310**

35.6.2 Generating Halftone Screens **311**

35.7 Halftone Dictionaries **312**

35.7.1 HalftoneType 1 Halftone Dictionaries **312**

35.7.2 HalftoneType 3 Halftone Dictionaries **314**

35.7.3 HalftoneType 5 Halftone Dictionaries **314**

35.8 Defining Halftone Functions **315**

35.9 Operators **315**

35.9.1 CIE Color Conversion Operators **315**

35.9.2 Device Color Conversion Operators **316**

35.9.3 Halftone Function Operators **316**

36 Exception Handling **317**

36.1 Structure Exceptions **317**

36.1.1 Syntax error in structure **317**

36.1.2 Illegal specification in structure **318**

36.1.3 Implementation constraint violation **318**

36.1.4 Unhandled content exception in content **318**

36.1.5 Unhandled structure exception in subordinate BLOCK **318**

36.2 Structure Exception Handling **318**

36.2.1 dpi::abort-policy::struggle-on **318**

36.2.2 dpi::abort-policy::on-error, dpi::abort-policy::on-warning **319**

36.3 Content Exceptions **319**

36.4 Content Exception Handling **320**

36.4.1 Handled exceptions **320**

36.4.2 Unhandled exceptions **320**

36.5 Content Exception Operators and Procedures **320**

36.5.1 RaiseException **320**

36.5.2 RaiseError **320**

36.5.3 RaiseWarning **321**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642478 563

36.5.4 Print ... 321

36.5.5 ExecuteTrapped ... 321

36.5.6 StoreErrorInfo ... 321

36.5.7 ReportErrorInfo ... 322

36.6 Interpreter Errors ... 322

36.6.1 Error Procedures and Dictionaries ... 322

36.6.2 Possible Interpreter Errors ... 323

36.7 ErrorInfoDict ... 324

36.7.1 newerror ... 324

36.7.2 errorname ... 324

36.7.3 command ... 324

36.7.4 ostack ... 324

36.7.5 dstack ... 325

36.7.6 recordstacks ... 325

36.8 Print requestor ... 325

37 Clear Text Representation and Interchange Format ... 325

37.1 Clear Text Structure Representation and Interchange Format ... 325

37.1.1 Base Structure Element Value Types ... 325

37.1.2 SPDL Document Type Definition ... 327

37.2 Clear Text Content Representation and Interchange Format 345

37.2.1 Token structure and delimiters ... 345

37.2.2 Token types ... 345

37.2.3 Encodings for content object types ... 345

37.2.4 In-line Raster Graphic Image Data Tokens ... 349

38 Binary Representation and Interchange Format ... 349

38.1 Binary Structure Representation and Interchange Format 349

38.2 Base Structure Element Value Types ... 350

38.2.1 Boolean type ... 350

38.2.2 Enumerated types ... 350

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642479 4TT

38.2.3 Integer type **350**

38.2.4 Real type **350**

38.2.5 Printable String type **350**

38.2.6 Octet String type **350**

38.2.7 Name type **350**

38.2.8 Object Identifier type **350**

38.2.9 Public Identifier type **350**

38.2.10 Environment Name type **350**

38.2.11 Structured Name type **350**

38.2.12 Comments **350**

38.3 SPDL Structure Definition Module **351**

38.4 SPDL Document Module **352**

38.5 External Reference Module **355**

38.6 Resource Definition Module **357**

38.7 Fonts and Glyph Index Map Module **359**

38.8 Document Production Instructions Module **362**

38.9 Binary Content Representation and Interchange Format **369**

38.9.1 Token structure and notation **369**

38.9.2 Token types **370**

38.9.3 Representation of values **371**

38.9.4 Short Opcode tokens **373**

38.9.5 Type/Value tokens **374**

38.9.6 Type/Length/Value tokens **374**

38.9.7 Short Integer tokens **375**

38.9.8 Encoding values for operators **375**

39 Conformance **375**

39.1 SPDL Instances **375**

39.2 SPDL Presentation Processes **376**

39.3 Representation and Interchange Format **376**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642480 111

39.4 Structure Syntax **376**

39.5 Resources **376**

39.5.1 Resource Definitions which are Subordinate to an ENVIRONMENT RESOURCE **376**

39.5.2 Resource Definitions which are Subordinate to a DOCUMENT **377**

39.5.3 Resources in the Environment of the Presentation Process **377**

39.5.4 Color Space Families **379**

39.6 External Structure Elements **380**

39.7 Document Production Instructions **380**

39.8 Content Processing **380**

39.8.1 Ability to construct and save objects **380**

39.8.2 Ability to represent values **381**

39.9 Exception Handling **381**

39.10 Imaging **381**

Annexes

A Operator Encodings **382**

B ASN.1 Object Identifiers and SGML Formal Public Identifiers defined by this International Standard **389**

B.1 Object Identifiers and SGML Formal Public Identifiers defined by this International Standard. **389**

B.1.1 Public identifiers **389**

B.1.2 ASN 1 Object Identifier Values **389**

B.1.3 Object Identifier Value Declarations Module **393**

C SPDL Indexed Font Representation Format **396**

C.1 Introduction **396**

C.2 Conventions **396**

C.3 Definitions **397**

C.4 SPDL Indexed Font Representation **397**

C.4.1 Header **397**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642481 058

C.4.2 PerFont Attributes **397**

C.4.3 Glyph Shape Information **401**

D SPDL mandatory font set for interchange **407**

E Glyph Index Maps **417**

E.1 GlyphIndexMap::Latin1Publishing **417**

E.2 GlyphIndexMap::Latin1PublishingA **426**

F ASN.1 Object Identifiers and Public Identifiers for Document Production Instructions defined by ISO/IEC 10175 ..... **435**

F.1 ASN.1 Object Identifiers and Public Identifiers for Document Production Instructions defined by ISO/IEC 10175 ..... **435**

F.1.1 Document Production Instruction Values and Meanings **435**

F.1.2 ASN.1 Object Identifiers and Public Identifiers **447**

F.1.3 Object Identifier Value References **447**

F.1.4 ASN.1 Object Identifier Definitions **458**

G Bibliography **464**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642482 T94

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 10180 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Annexes A, B, C, D and E form an integral part of this International Standard. Annexes F and G are for information only.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642483 920

Introduction

The development of laser printers in the early 1970's provided the opportunity for the low cost printing of documents containing not only character text but also general graphical material. This has in turn resulted in increasing interest in a standard printer interface which would make the capabilities of these printers available to document creation systems. Initial efforts to provide such an interface have in general taken two directions: the enhancement of existing printer interfaces to take advantage of additional printer capabilities, and the development of new document description technologies which are more appropriate to the new printing technology's enhanced imaging capabilities. The most successful of these efforts involved a departure from techniques and standards existing at that time for printing text in favor of techniques based on the model of computer languages. These printer interfaces have come to be known as "Page Description Languages" or "PDLs".

This International Standard specifies a device-independent means of describing documents comprised of text and graphical material, for presentation on paper or other media. SPDL also allows processes other than the presentation process to select pages or portions of SPDL documents for their use.

This International Standard is organized as follows:

a) The general architecture of SPDL is specified.

b) The major components and semantics of SPDL structure are specified.

c) The interface between structure processing and content processing is specified

d) The major components and semantics of the SPDL content notation are specified.

e) Exception handling in both structure processing and content processing are specified.

f) Two interchange formats are specified.

g) Statements covering both document conformance and presentation process conformance are provided.

h) Normative annexes are provided which specify information objects which are defined by this International Standard and a font object format.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642484 867

i) An informative annex is provided to identify information objects defined by ISO/IEC 10175 which are included in this International Standard by reference.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642485 7T3

© ISO/IEC

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0642486 63T

Information technology - Processing languages - Standard Page Description Language (SPDL)

1 Scope

1.1 Scope

This International Standard defines a language for the specification of electronic documents, comprised of black and white, gray scale, or full color text, images, and geometric graphics, in a form suitable for presentation (printing or displaying on other suitable media).

This International Standard is intended to be extensible in order to accommodate future developments in imaging technology

This International Standard is intended to be used in a variety of configurations meeting a variety of connectivity needs. It is specifically compatible with use over OSI networks.

In addition to specifying how document images are represented, this International Standard specifies how Document Production Instructions affect document presentation.

1.2 Field of Application

This International Standard is intended for use in a wide variety of printing and publishing environments, including:

— electronic publishing;

— office systems;

— information networks;

— demand printing.

Documents conforming to this International Standard are termed SPDL documents. SPDL documents may be:

— interchanged;

— processed by locally connected presentation devices;

— sent to presentation devices connected to OSI or non-OSI networks;

— stored for presentation at a later time

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

1.3 Relationship to Other Standards

The encoding of the document structure of documents conforming to this International Standard conforms to ISO/IEC 8824 and ISO/IEC 8825 if the binary encoding is used or to ISO/IEC 8879 if the clear text encoding is used.

This International Standard provides a detailed specification of the effect produced when a font resource conforming to the architecture of ISO/IEC 9541 is used in the presentation of character text. Structured Names as used in this International Standard for glyph identification are as defined in ISO/IEC 9541-2 and ISO/IEC 9070.

This International Standard provides a straightforward and efficient way of representing documents which are generated for printing by ODA systems. It also provides a capability of representing documents generated by SGML applications whose formatting is described by ISO/IEC 10179.

This International Standard provides for the efficient implementation of Print Services conforming to ISO/IEC 10175 which incorporate an SPDL Presentation Process for document presentation. This International Standard also specifies the specific semantics of applicable Document Production Instructions.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards

ISO/IEC 646:1991, *Information technology - ISO 7-bit coded character set for information interchange.*

ISO/IEC 8824:1990, *Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8825:1990, *Information technology - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO 8879:1986, *Information processing - Text and office systems - Standard Generalized Markup Language (SGML).*

ISO/IEC 9070:1991, *Information technology - SGML support facilities - Registration procedures for public text owner identifiers.*

ISO/IEC 9541-1:1991, *Information technology - Font information interchange - Part 1: Architecture.*

ISO/IEC 9541-2:1991, *Information technology - Font information interchange - Part 2: Interchange Format.*

ISO/IEC 9541-3:1994, *Information technology - Font information interchange - Part 3: Glyph shape representation.*

ISO/IEC 10175-1[1], *Information technology - Text and office systems - Document printing application - Part 1: Abstract service definition and procedures.*

ISO/IEC 10175-2[2], *Information technology - Text and office systems - Document printing application - Part 2: Protocol specification.*

IEEE 754:1985, *IEEE Standard for Binary Floating-Point Arithmetic.*

1) To be published

2) To be published

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642488 402

3 Definitions

For the purposes of this International Standard, the following definitions apply

3.1 Additional DPI: Document Production Instructions which are not defined by this International Standard.

3.2 AFII Glyph Name type: The subset of the Simple Glyph Name type consisting of names of the form "afii*nnnn*" where "*nnnn*" is a sequence of decimal digits beginning with a non-zero digit which represent an integer in the range of 1 to 2^{31}-1.

3.3 base font: A font which contains actual glyph representations; in a composite font, a leaf node in the composite font tree structure.

3.4 base structure element: A structure element which cannot be subdivided into smaller structure elements.

3.5 bind: To associate a specific object with an Identifier used to identify that object.

3.6 bit: A representation of a unit of information which may have the value of *zero* or *one*.

3.7 Block State: A set of attributes associated with the processing of a BLOCK at a particular point in time.

3.8 Boolean type: A type with two values: *True* and *False*.

3.9 Cardinal type: A subtype of the Integer type whose values are the non-negative integers

3.10 character text: Text which is presented by means of one or more sequences of glyph images.

3.11 clip: To remove a portion of a picture or graphic symbol by not rendering the removed portion.

3.12 clipping region: An element of the imaging model which restricts the region of the Current Page Image on which ink may be placed. Any portion of a graphic image which lies outside the clipping region will be clipped.

3.13 composite font: A hierarchically organized (tree structured) collection of fonts.

3.14 composite structure element: A structure element which can be subdivided into smaller structure elements.

3.15 Composition and Layout Process: The element of the Document Processing Model which is responsible for the composition of character text, the formatting of character text and graphics, and the positioning of character text and graphics for presentation.

3.16 content *(of a composite structure element)*: The content of all of the TOKENSEQUENCE structure elements which are subordinate to the composite structure element

3.17 content *(of a TOKENSEQUENCE)*: The value of the TOKENSEQUENCE structure element.

3.18 content element: An octet string which represents a sequence of tokens in the Content Representation and Interchange Format.

3.19 Content Processor: The element of the Document Processing Model which is responsible for the processing of the document content.

3.20 content value: A content element together with information which specifies its encoding.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.21 Context Dictionary: A Dictionary which is defined by a Dictionary RESOURCE DEFINITION structure element.

3.22 Context of Interpretation: The context in which a content of a TOKENSEQUENCE is interpreted by the Content Processor.

3.23 Context Stack: Part of the Virtual Machine, consisting of an ordered set of Dictionaries in which name look-up occurs.

3.24 Creation and Editing Process: The element of the Document Processing Model which is responsible for the creation and editing (modification) of the document

3.25 Current Context of Interpretation: A Context of Interpretation which is associated with a particular Block State and which is used for the processing of the content of subordinate TOKENSEQUENCE structure elements

3.26 Current Page Image: The element of the Block State of a PAGE in which the page image for that PAGE is constructed

3.27 descendant font: In a composite font, a font at a lower level of the hierarchy than the root font.

3.28 Dictionary: A data structure constructed by content processing consisting of name/value pairs.

3.29 document: When not otherwise qualified, the electronic form of information which may be rendered in visible form.

3.30 document creator: A process that generates a final form document Unless otherwise qualified, the final form document is assumed to be an SPDL document.

3.31 Document DPI: The set of Document Production Instructions specified within the document.

3.32 Document Production Instructions: Instructions which affect the presentation of a document.

3.33 document structure: The set of components used to construct an SPDL document and their relationships.

3.34 Enumerated type: A type with an enumerated set of values.

3.35 environment (of the Presentation Process): The aggregate of all external information available to a Presentation Process.

3.36 Environment Identifier type: A type whose values include all identifiers used to identify information objects in the environment of the Presentation Process. The values of the Environment Identifier type consist of the values of the Public Identifier type, the Object Identifier type, and the Environment Name type.

3.37 Environment Name type: A type whose values are simple names which are used to identify information objects in the environment of the Presentation Process. The values of the Environment Name type consist of the set of object-name components as defined by ISO/IEC 9070.

3.38 External Structure Element: A structure element which is external to the document itself, but which is included in the document by reference

3.39 final form document: A document which is in a form suitable for presentation. As used in this International Standard, this term implies that all composition and layout decisions have been made. (See also "revisable form document".)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642490 060

NOTE 1 Final form generally implies that the document is no longer in a form that is amenable to being edited. However, revisable form and final form are not mutually exclusive concepts. Some data formats, for example ISO/IEC 8613 (ODA), allow both revisable form and final form information to be present in a single document at the same time.

3.40 Finishing Instruction: A Document Production Instruction which affects the document production process subsequent to the rendering of the page images.

3.41 Font Index Map: In a composite font, the Vector used to associate font indices with font selectors during glyph mapping.

3.42 Font Resource: A collection of glyph representations together with descriptive and font metric information which are relevant to the collection of glyph representations as a whole.

3.43 glyph: A recognizable abstract graphic symbol which is independent of any specific design. (See ISO/IEC 9541-1)

3.44 Glyph Identifier type: A type whose values consist of the values of the Simple Glyph Name type and the Structured Name type.

3.45 glyph image: An image of a glyph, as obtained from a glyph representation, which is displayed on a presentation surface.

3.46 glyph index map: A mapping of glyph index values to the glyph identifiers which identify the glyph representations in a Font Resource.

3.47 glyph representation: The glyph shape and glyph metrics associated with a specific glyph in a font resource.

3.48 glyph shape: The set of information in a glyph representation used for defining the glyph image.

3.49 graphics state: A set of values for the graphics state variables

3.50 graphics state stack: An ordered set of saved graphics states.

3.51 graphics state variables: A set of variables which are used in conjunction with the explicit arguments to an imaging operator to specify the change to the page image which results from the execution of the imaging operator.

3.52 highest structure level: A structure element which is not subordinate to any other structure element is said to be at the highest structure level.

3.53 Identifier type: A type whose values are used to identify objects in an instance of SPDL.

3.54 imageable region: The region of the Current Page Image upon which imaging can occur.

3.55 Imager: The component of the Content Processor which executes imaging operators and which makes changes to the Current Page Image.

3.56 imaging: The process of generating the document page images by making changes to an initially blank Current Page Image as specified in clause 17.

3.57 imaging activity: Painting a mask clipped by the *CurrentClippingRegion* onto the Current Page Image as specified in clause 17.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.58 imaging model: The model which defines how imaging operators affect the state of the Current Page Image

3.59 imaging operator: An operator whose execution causes a change to the Current Page Image.

3.60 immediate subordinates: The parts into which a composite structure element has been subdivided.

3.61 implementation *(of SPDL)*: An actual system which presents SPDL documents.

3.62 imposition: The placement of the images of two or more logical pages on the same side of an instance of a sheet medium (such as paper). Imposition is the responsibility of the Composition and Layout Process.

3.63 in-line data: Raster graphic image data which is physically contained within an SPDL document (as opposed to external data which is logically contained with the document).

3.64 Indexed Font: A *combination of a Font Resource and a Glyph Index Map, which is used for the presentation of character text.*

3.65 information object: An object in the environment of the Presentation Process, which represents a structured or unstructured unit of information. Information objects include values, resources and includable structure elements.

3.66 ink: An element of the imaging model which specifies the color and visual texture which will be displayed on the presentation device.

3.67 instance of SPDL: A document, resource, or includable structure element represented in the Standard Page Description Language defined in this International Standard.

3.68 Integer type: *A type whose values are a subset of the mathematical integers.*

3.69 ISO10036 Glyph Name type: A subset of the Structured Name type consisting of those Structured-names for which the owner-name identifies the registration authority authorized by ISO/IEC 10036.

3.70 logical page: A part of a document which may be recognized and managed by a Composition and Layout Process. Logical pages have no explicit representation in SPDL document structure

NOTE 2 Logical pages are normally represented by PICTURE structure elements.

3.71 Management Instructions: Document Production Instructions which affect the management of the document production process.

3.72 mask: *An element of the imaging model which provides the shape of an image element*

3.73 medium: The physical surface upon which a document is rendered in visible form by a presentation device. Examples include paper and the display screens of video display terminals.

3.74 Medium Declarations: Document Production Instructions which specify media which may be used for the presentation of the document

3.75 Name type: A type whose values are sequences of characters as defined in 18.1.8.

3.76 Non-Negative Number type: A type whose values are non-negative values of the Number type.

3.77 Number type: A type whose values are the values of the Integer type and the Real type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642492 933

3.78 object: One of the data objects which can be operated upon by the Virtual Machine. See also *information object*.

3.79 Object Identifier type: A type whose values consist of object identifiers as defined by ISO/IEC 8824.

3.80 Object Name: A name which identifies an information object to which this International Standard assigns both an object identifier value as defined by ISO/IEC 8824 and a public identifier as defined by ISO/IEC 9070.

3.81 object type: One of the identified object types associated with SPDL virtual machine objects.

3.82 octet: An ordered sequence of eight bits

3.83 Octet String type: A type whose values are ordered sequences of zero or more octets.

3.84 operator: A token the interpretation of which causes the Virtual Machine to perform an operation.

3.85 page description language: One of a group of print formats based on the concepts of programming languages.

3.86 page image: An abstraction of the medium upon which presentation is to occur.

3.87 parent font: In a composite font, a parent font is one which has a given descendant font as a component of the sub-tree for which the parent font is the root node.

3.88 parser: The part of the Virtual Machine which decomposes a content value into a sequence of tokens.

3.89 path: A conceptual sequence of zero or more points, lines, and/or curves defining geometry in the Reference Coordinate System.

3.90 peer (structure elements): Structure elements which are immediately subordinate to the same structure element

3.91 picture: An image which can be considered in isolation from other images, e.g. a page or a diagram on a page.

3.92 Positive Integer type: A sub-type of the Integer type whose values are the positive integers.

3.93 Pre-processing Instruction: A Document Production Instruction which affects the document production process prior to the rendering of the document page images

3.94 presentation: The rendering of a document in human-perceivable form; in the context of this International Standard, the rendering of an electronic document in visible form.

3.95 presentation device: A physical device which presents a document in visible form.

3.96 Presentation Instruction: A Document Production Instruction which affects the process of rendering the document page images and their placement on the media.

3.97 presentation order: The sequential order of structure elements of the SPDL document.

3.98 presentation process: A process which renders a document, or portion thereof, on suitable media.

3.99 Printable String type: A type whose values are those of the PrintableString type as defined in ISO 8824.

3.100 Public Identifier type: A type whose values are public identifiers as defined by ISO/IEC 9070.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.101 Public Object Identifier type: A type whose values are public identifiers as defined by ISO/IEC 9070 and object identifiers as defined by ISO/IEC 8824.

3.102 Real type: A type whose values are those mathematical rational numbers which can be represented as specified by IEEE 754.

3.103 Reference Coordinate System (RCS): A Cartesian coordinate system with units of millimeters on both axes which provides unambiguous specification of locations on the medium.

3.104 rendering: the creation of a visible image of a page image or a document as part of the presentation process.

3.105 resource: An information object which may be available in the environment of the Presentation Process and which may be used by reference in the presentation of an SPDL document

3.106 revisable form document: A document which is in a form suitable for editing and revising by some properly equipped system. This term generally implies that, in addition to its explicit content, the document contains information which identifies its logical structure. (See also *final form document*.)

3.107 root font: In a composite font, the font at the top level of the hierarchy.

3.108 SaveObject: An object which represents the state of the Virtual Machine at a particular point in time.

3.109 Simple Glyph Name type: A type whose values are a subset of the values of the Name type.

3.110 SPDL document: A (final form) document represented in the Standard Page Description Language defined by this International Standard.

3.111 SPDL Presentation Process: A process which presents an SPDL document in visible form; part of the basic Document Processing Model.

3.112 SPDL structure: The set of components used to construct an instance of SPDL and their relationships.

3.113 SPDL Virtual Machine: The virtual machine in terms of which SPDL Content Processing is specified.

3.114 state variables: A set of variables which are part of the Virtual Machine state.

3.115 structure element: Any of the elements of an instance of SPDL which are distinguished by the SPDL structure.

3.116 structure processing: The processing of the set of components used to construct an SPDL document.

3.117 Structure Processor: The part of the SPDL document processing model which is responsible for processing the SPDL structure.

3.118 Structured Name: A form of public identifier which is defined by ISO/IEC 9541-2. Structured Names are semantically equivalent to public identifiers as defined by ISO/IEC 9070. They are used for identifying glyphs and elements of Font Resources.

3.119 subordinate: A structure element which results from the further subdivision of another structure element shall be said to be subordinates of or subordinate to that structure element.

3.120 superior: A structure element A is said to be superior to a structure element B if B is subordinate to A.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642494 706

3.121 Supplementary DPI: Document Production Instructions which are associated with a particular instance of document presentation but which are not contained within the document itself.

3.122 tiling: The process of laying down copies of a pattern to fill an area.

3.123 token: A logical unit of information which is processed by the SPDL Virtual Machine. The processing of each token causes a specified change to the Virtual Machine state.

3.124 User Coordinate System (UCS): A dimensionless Cartesian coordinate system which specifies locations in the Current Page Image by means of a mapping to the Reference Coordinate System defined by the *CurrentTransformation*.

3.125 virtual machine: An abstraction which allows the specification of a process in terms of changes to the state of an idealized implementation. Such an idealized description serves only to specify the process, not to dictate the actual implementation.

3.126 Virtual Machine State: The set of attributes belonging to and characterizing the Virtual Machine at a particular point in time.

4 SPDL General Architecture

This clause describes the general architecture of the Standard Page Description Language.

NOTE 3 In the following clauses, objects and structures which are specific to this International Standard are named by proper names (indicated by names beginning with capital letters), while generic objects are indicated by common names (indicated by names beginning with small letters).

4.1 Document Processing Model

The basic document processing model consists of three processes and two forms of document representation as illustrated in figure 1.

The *Creation and Editing Process* generates the character text and graphics which comprise the logical content of the document, and the logical document structure. This information is represented as a *revisable form document*. The *Composition and Layout Process* formats the character text and graphics contained in the revisable form document for presentation, and incorporates the composition and layout decisions in a *final form document*. The *Presentation Process* renders the final form document in visible form. These processes are further described below

NOTE 4 This model is of necessity greatly simplified. Actual document processing is usually highly iterative. Revisable form documents are formatted for proofing, and the results of the Composition and Layout Process must be presented to be evaluated. Many applications also include multiple processes. Document editing systems frequently include a Composition and Layout and Presentation Process for document previewing, while some printers may combine a Composition and Layout Processes with the Presentation Processes

4.1.1 Creation and Editing Process

The Creation and Editing Process is responsible for the creation of the character text and graphics which comprise the logical content of the document and for the logical structure of the document. It may also generate information which specifies how the Composition and Layout Process is to format the document for presentation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642495 642

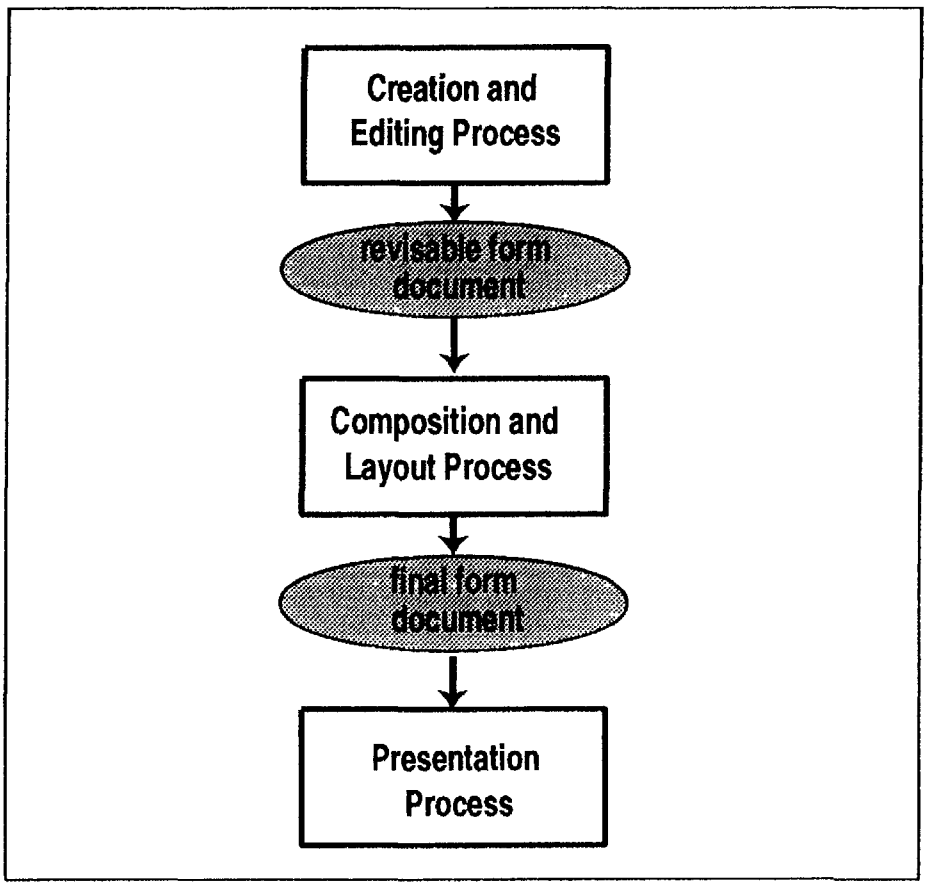

Figure 1 — Document Processing Model

4.1.2 Revisable Form Document

The revisable form document represents the logical content and structure of the document created by the Editing Process. It also includes any information generated by the Editing Process which specifies how the document is to be formatted for presentation.

4.1.3 Composition and Layout Process

The Composition and Layout Process composes character text of the document, formats any graphical elements, and lays out the composed character text and graphics for presentation. All composition, formatting, and positioning decisions are made by the Composition and Layout Process. These decisions include such details as

— selection of glyphs used to present character text;

— positioning of each glyph, either implicitly by reference to font metrics or explicitly;

— choice of line width, line end, and line join characteristics for graphics;

— choice of fill patterns for geometric objects.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642496 589

The formatting specifications which direct the Composition and Layout Process may be part of the revisable form document or they may be separate from the document. ISO/IEC 10179 defines one form for formatting specifications which are not part of the revisable form document.

In addition to the revisable form document and formatting specifications, input to the Composition and Layout Process may include previously formatted elements which are to be incorporated in the final form document.

4.1.4 Final Form Document

The final form document represents the results of the composition, formatting, and positioning decisions made by the Composition and Layout Process. It is not required to represent the logical document structure, the formatting specifications, or the intent of those decisions.

4.1.5 Presentation Process

The Presentation Process presents an document by rendering it in visible form on suitable media. It also performs any finishing operations (e.g. trimming, stapling, or binding) which are part of document presentation.

4.2 SPDL Document

SPDL is a final form document format. As such it provides facilities for the efficient representation of the decisions made by the Composition and Layout Process. SPDL does not provide facilities for the representation of logical document structure.

4.2.1 Sources of SPDL Documents

The primary source of SPDL documents is the application of a Composition and Layout Process to a revisable form document. In addition, an SPDL document may also be created by:

— translation of a fully formatted document represented in some alternate final form document format;

— an imposition process or other Composition and Layout Process applied to final form document or to a combination of final form and revisable form documents.

4.2.2 SPDL Document Presentation

The SPDL document represents the ideal image specified by the Composition and Layout Process. It is the responsibility of the SPDL Presentation Process to adapt the rendering of the document to the constraints of the presentation device so as to provide the best possible approximation to the description specified by the Composition and Layout Process.

An SPDL document may be presented many times, at different times and/or places. For this reason document presentation must accommodate parameters which may be specific to a particular instance of presentation. These parameters are specified by *Document Production Instructions*.

Document Production Instructions may be conveyed to the Presentation Process as part of the SPDL document or made available via a separate mechanism. The method by which Document Production Instructions which are not a part of the document are made available to the Presentation Process is outside the scope of this International Standard.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642497 415

4.2.3 Uses of SPDL Documents

The primary use of SPDL documents is input to an SPDL Presentation Process. In addition, an SPDL document may be used to represent a formatted portion of document content. Such previously formatted document content may be used as input to a Composition and Layout Process for inclusion in another document.

4.3 SPDL Document Architecture

An SPDL document has *structure* and *content*. The structure organizes the content into separable parts and identifies the resource requirements for each part. The structure is self contained and can be processed without processing the content.

The content is in the form of a page description language which specifies how the page images are to be generated. The processing of the content is dependent on the processing of the structure: the processing of the structure establishes the context in which the content can be processed.

The Presentation Process may also have available to it various sources of additional data, such as fonts or stored portions of SPDL documents, which may be referenced by an SPDL document. The sum of all such data is called the *environment* of the Presentation Process.

4.3.1 Document Structure

The document structure is the result of the division and repeated subdivision of a document into increasingly smaller parts. The resulting parts are called *structure elements*. An example of the document structure is shown in figure 2.

The document structure is hierarchical. The highest level structure element is a DOCUMENT. A DOCUMENT may contain subordinate PAGESET and/or PAGE structure elements.

A PAGESET may in turn contain subordinate PAGESET and/or PAGE structure elements. Each PAGE contains the description of that portion of the document which is to be presented on a single instance of the presentation medium.

A PAGE may contain subordinate PICTURE and/or TOKENSEQUENCE structure elements. A PICTURE may contain subordinate PICTURE and/or TOKENSEQUENCE structure elements. A TOKENSEQUENCE is a structure element which contains document content.

DOCUMENT and PAGE structure elements are semantically distinct but are not syntactically distinguished. A DOCUMENT is simply a PAGESET or a PICTURE which is at the highest level in the structure hierarchy. Similarly, a PAGE is simply a PICTURE which is at the highest level in the PICTURE hierarchy.

In addition to representing the division of an document into smaller parts, the document structure supports:

— the sequencing of pages of the document in the correct presentation order;

— the application of Document Production Instructions;

— the establishment of a context of interpretation for content elements;

— resolution of references to structure elements which are external to the document;

— resolution of references to resources which are external to the document;

— the specification of resources;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

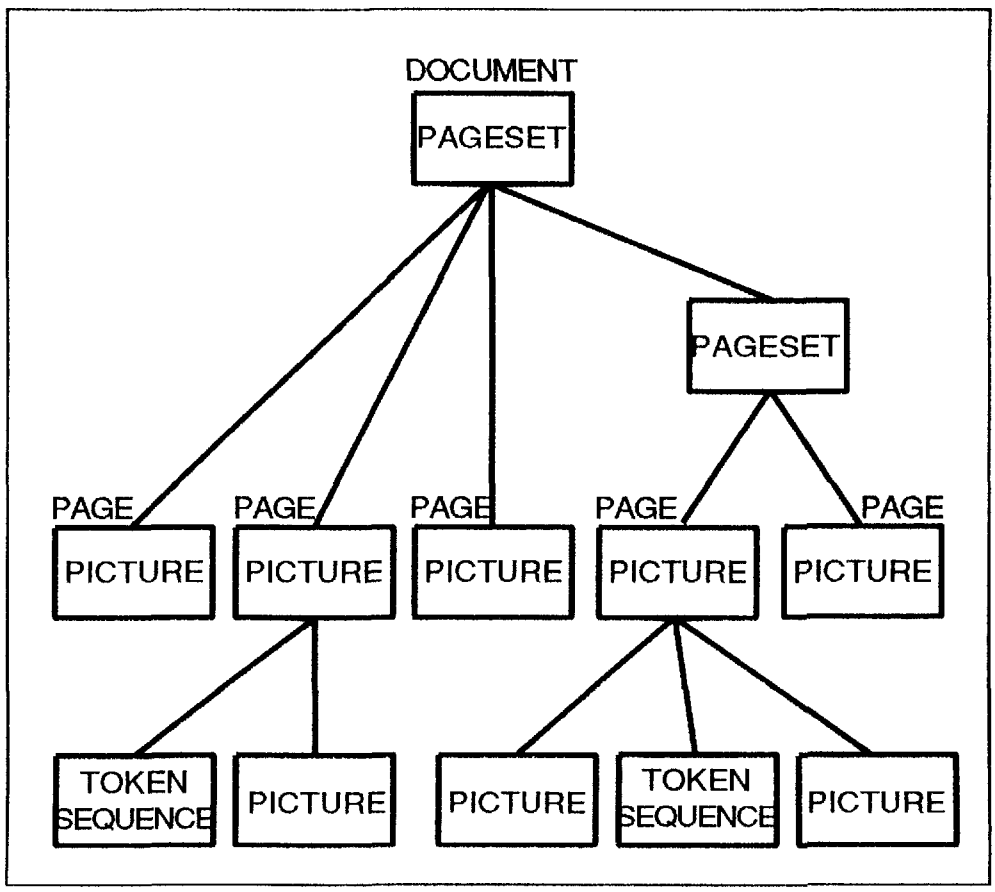

Figure 2 — Example of SPDL Document Structure

— the selection of pages or portions thereof by other processes.

NOTE 5 Processes which process SPDL structure for purposes other than document presentation (e.g. a Composition and Layout Process which extracts PICTURE structure elements for inclusion in other documents, or an Imposition Processes which produces a new document by combining multiple page images on a single sheet) are outside the scope of this International Standard.

4.3.2 Document Content

The *content* of a document is the data contained in TOKENSEQUENCE structure elements. Document content within each TOKENSEQUENCE consists of a sequence of tokens Document content processing consists of interpreting the sequence of tokens in each TOKENSEQUENCE. The overall process of token interpretation is modeled as a virtual state machine.

4.3.3 External structure elements

An *External Structure Element* is a structure element which is included in an SPDL document by reference. An SPDL document that includes an External Structure Element by reference must declare it in an EXTERNAL DECLARATION structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4.3.4 Resources

A *resource* is an information object which is available in the environment of the Presentation Process. Resources may be defined as specified in this International Standard or may be made available in the environment of the Presentation Process by means of a processes which is outside of the scope of this International Standard. A resource may be used, by reference, within the document content of an SPDL document. An SPDL document that references a resource must declare it in a RESOURCE DECLARATION structure element.

4.3.5 Document Presentation

Since the SPDL document consists of document structure and document content, and since these categories of data are separable, the Presentation Process can be modeled as containing subsidiary processes which process structure and content separately. In this International Standard, these processes are referred to as the *Structure Processor* and the *Content Processor* respectively. The Presentation Process also contains *Marking and Finishing Process* which renders the page images on the medium and performs any document finishing processes. See figure 3).

NOTE 6 While structure processing and content processing are logically separable processes, separate processes need not exist within an implementation of the Presentation Process. The only requirement is that presentation of SPDL documents occurs with the results specified in this International Standard.

Document structure processing consists of analyzing the structure elements of the SPDL document, their types, their hierarchical relationships, and the presentation order. For presentation, structure processing also:

— resolves interactions between Document Production Instructions;

— applies Document Production Instructions to appropriate PAGESETs and PAGEs of the SPDL document,

— establishes the context of interpretation for each TOKENSEQUENCE

Document content processing consists of interpreting each content element as a sequence of tokens in the Content Representation and Interchange Format and processing those tokens as specified in clause 17 through clause 36.

Document presentation includes all of the artifacts of the Presentation Process, including the generation of error messages and warnings.

4.3.6 Representation and Interchange Format

SPDL documents are interchanged in the Representation and Interchange Format defined in clause 37 and clause 38. The Representation and Interchange Format has two forms, a binary form and a clear text form, and two parts, a Structure Representation and Interchange Format and a Content Representation and Interchange Format. The binary form of the Content Representation and Interchange Format is used with the binary form of Structure Representation and Interchange Format. The clear text form of the Content Representation and Interchange Format is used with the clear text form of the Structure Representation and Interchange Format.

The binary form of the Structure Representation and Interchange Format uses Abstract Syntax Notation 1 (ASN.1) as defined in ISO/IEC 8824 encoded using the Basic Encoding Rules specified by ISO/IEC 8825. The clear text form of the Structure Representation and Interchange Format uses Standard Generalized Markup Language (SGML) as defined in ISO 8879 encoded using ISO/IEC 646 IRV.

The Content Representation and Interchange Format uses specialized encodings optimized for compactness and efficiency for each of the binary and the clear text forms.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

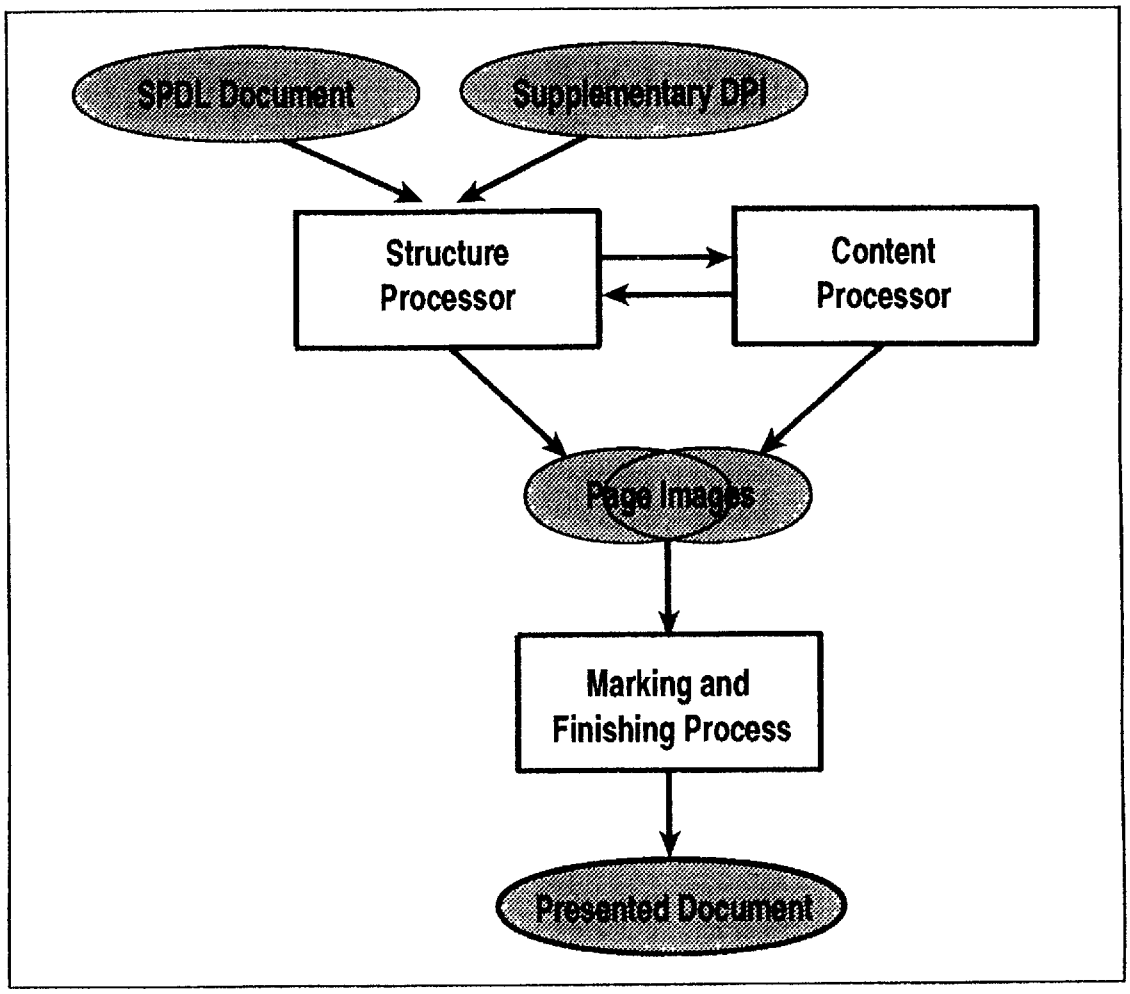

Figure 3 — SPDL Presentation Process

4.4 Relationship to Print Services

A Print Service which has the capability of printing SPDL documents has an SPDL Presentation Process as part of the Print Service. This clause discusses the relationship of such a Print Service to the embedded SPDL Presentation Process.

4.4.1 Document Production Instructions

When an SPDL document is printed by a Print Service, the print operation parameters may include parameters which affect document imaging and presentation. Such parameters specify Document Production Instructions which are called Supplementary DPI The Print Service implementation is responsible for making the SPDL document and Supplementary DPI available to the SPDL Presentation Process.

The SPDL Presentation Process resolves interactions between Document Production Instructions which are provided as Supplementary DPI associated with a particular instance of presentation and any Document Production Instructions contained in the SPDL document. This relationship is illustrated in figure 4.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

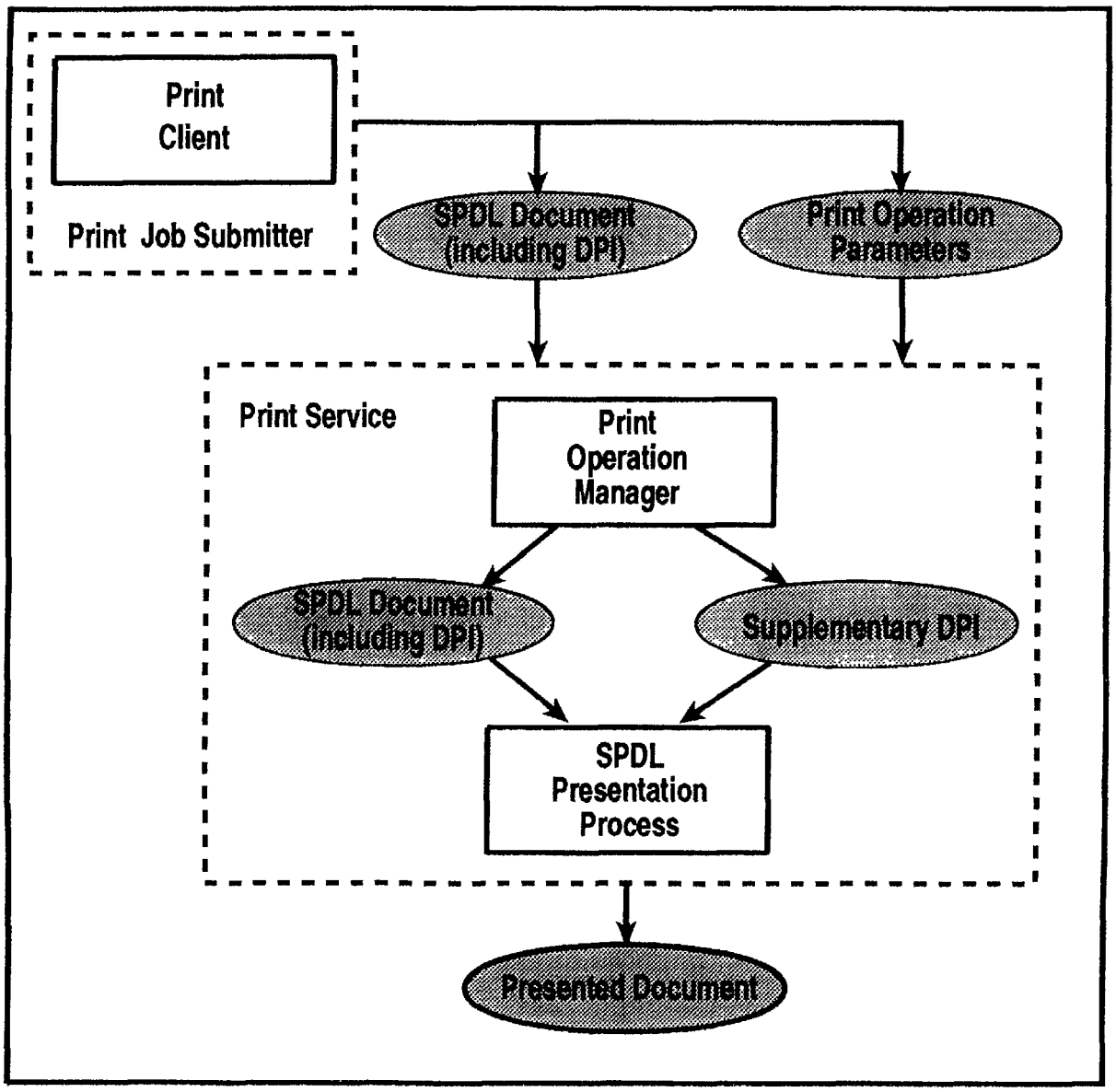

Figure 4 — Relationship of SPDL Documents to a DPA Print Service

4.4.2 Relationship to ISO/IEC 10175

ISO/IEC 10175 defines a standard Document Printing Application which provides a standard (abstract) Print Service. Document printing is accomplished by the invocation of a Print Operation accompanied by a document and a set of Print Operation Parameters.

The Print Operation Parameters defined by ISO/IEC 10175 consist of Print Job Management Instructions and Document Production Instructions.

Print Job Management Instructions do not affect the presentation of SPDL documents. Some, but not all, of the Document Production Instructions defined by ISO/IEC 10175 are applicable to an SPDL Presentation Process.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642502 602

The semantics of ISO/IEC 10175 Print Job Management Instructions and the general semantics of those Document Production Instructions which may occur as ISO/IEC 10175 Print Operation Parameters are defined by ISO/IEC 10175. The specific semantics of those Document Production Instructions which may occur as ISO/IEC 10175 Print Operation Parameters and which are applicable to the presentation of SPDL documents, when used in the presentation of SPDL documents, are defined by this International Standard.

5 SPDL Structure

The SPDL structure is the result of the division and repeated subdivision of an SPDL instance into increasingly smaller parts. This process results in a hierarchical structure.

The notation employed to specify the SPDL structure is declarative in nature, as opposed to the content notation, which takes the form of a procedural language. The structure also implies a sequential order for the set of structure elements which comprise the document.

5.1 Hierarchical structure

The following concepts and terminology are associated with the hierarchical structure and the relationships implied by that structure.

5.1.1 Structure elements

The elements of an SPDL document which are distinguished by the document structure are called *structure elements*.

5.1.2 Composite structure elements

A structure element which is subdivided into smaller structure elements is called a *composite structure element* For each composite structure element, the defining clause specifies which subordinate structure elements may occur.

5.1.3 Base structure elements

A structure element which cannot be subdivided into smaller structure elements is called a *base structure element.*

Every base structure element has a value. The value of a base structure element belongs to one of the types defined in clause 6. For each base structure element, the defining clause specifies the type of the value of the structure element.

5.1.4 Subordinates and immediate subordinates

The parts into which a composite structure element has been subdivided are said to be the *immediate subordinates* of the composite structure element.

Since the immediate subordinates of a composite structure element may be composite structure elements, there may be structure elements which are the result of the further subdivision of a composite structure element which are not themselves immediate subordinates of that structure element. All of the structure elements, composite or base, which result from the further subdivision of a structure element are said to be *subordinates of* or *subordinate to* that structure element.

Reference to the immediate subordinates of a structure element always uses the form "*immediate subordinates*" or "*immediately subordinate*". When the term "*subordinate*" is used without this qualifier it means subordinate to any hierarchical level.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5.1.5 Superiors and immediate superiors

It is often necessary to refer to the structure element or structure elements to which a particular structure element is subordinate. A structure element A is said to be *superior to* a structure element B (or *a superior of* B) if B is subordinate to A. A structure element A is said to be *immediately superior to* a structure element B if B is immediately subordinate to A.

Reference to the immediate superior of a structure element always uses the form "*immediate superior*" or "*immediately superior*". When the term "*superior*" is used without this qualifier it means a structure element which is superior to any hierarchical level.

5.1.6 Most immediately superior structure elements

In describing the structure or processing of a structure element *A*, the phrase *the most immediately superior X*, means the structure element *B* for which

— *B* is a structure element of class *X*

— *B* is superior to *A*

— there is no other structure element of class *X* which is both superior to *A* and subordinate to *B*.

5.1.7 Peers

Structure elements immediately subordinate to the same structure element are called *peers*. Structure elements which are neither subordinate nor superior to each other, and which do not share the same immediate superior, are *unrelated*.

5.1.8 Highest structure level

A structure element which is not subordinate to any other structure element is said to be *at the highest structure level*.

5.1.9 Document Structure Elements

A structure element which is subordinate to a DOCUMENT is called a *Document Structure Element*. A structure element of class *X* which is a Document Structure Element is called a *Document X*.

5.1.10 Scope

The *scope of a Document Structure Element* is the part of the document which can be affected by it. The end of the scope of a Document Structure element is the end of the most immediately superior BLOCK. The scopes of two Document Structure Elements which affect the same aspect of document processing, such as two Document Production Instructions, may overlap. The effects of overlapping scope are implied by the structure processing specifications.

The *scope of the binding of an Identifier to a resource or an external structure element* is the part of the document in which the Identifier can be used to reference the resource or external structure element to which it is bound. The end of the scope of the binding of an Identifier to a resource or an external structure element by a structure element is the end of the most immediately superior BLOCK. The scopes of two such bindings may overlap. The effects of overlapping scope are implied by the structure processing specifications.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642504 485

5.2 Sequential Order

In addition to a hierarchical structure, the SPDL structure assigns a sequential order to the immediate subordinates of each of the composite structure elements.

The combination of the hierarchical structure and the sequential order of peer structure elements generates a sequential order for the set of structure elements which comprise the SPDL INSTANCE. Structure elements that come later in this sequential order are said to *follow* the structure elements that come earlier, while the earlier structure elements are said to *precede* later structure elements

5.2.1 Sequential Order and Hierarchical Structure

The sequential order is related to the hierarchical structure by two rules:

— Each structure element is followed by all of its subordinates

— Any structure element preceded by a peer, is also preceded by all subordinates of that peer.

Thus the sequential order has the property of being "pre-order depth-first" relative to the hierarchical structure.

5.2.2 Example

An example of the relationship of the sequential order to the hierarchical structure is shown in figure 5.
In this example the structure elements are indicated by boxes, with subordinates being indicated by connecting lines. The order of the immediate subordinates of each structure element is left-to-right as indicated by the arrows The sequential order for the structure elements is identified by the numbers in the boxes.

5.2.3 Presentation Order

This sequential order of structure elements is also the *presentation order* of an SPDL document. This order is the order in which the Structure Processor processes the structure elements in the presentation of the document.

The term presentation order may be used to refer to the sequential order of a subset of the structure elements within an SPDL document. For instance, the "presentation order of PAGE structure elements" within the SPDL document is the order in which PAGE structure elements occur within the sequential order of the SPDL document. The presentation order of PAGE structure elements is the order in which the pages of the document are presented to the user.

5.2.4 Ordinal Page Number

Each of the PAGE structure elements is assigned an *ordinal page number* which identifies the PAGE structure element for the purposes of document presentation. The ordinal page numbers are assigned in the presentation order of the PAGE structure elements within the document.

5.2.5 Pages and Imposition

An imposition process may place two or more "logical pages" on a single sheet. Such an imposition process is part of the Composition and Layout Process. In the resulting SPDL document, each PAGE structure element describes the image to be placed on an entire sheet, which may include several "logical pages". The SPDL Presentation Process has no knowledge of the component logical pages, but only of the PAGE structure elements and the associated page images. The ordinal page numbers used in document presentation refer to the PAGE structure elements, and have no relationship to any page numbers which may be associated with the "logical pages" of the document.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

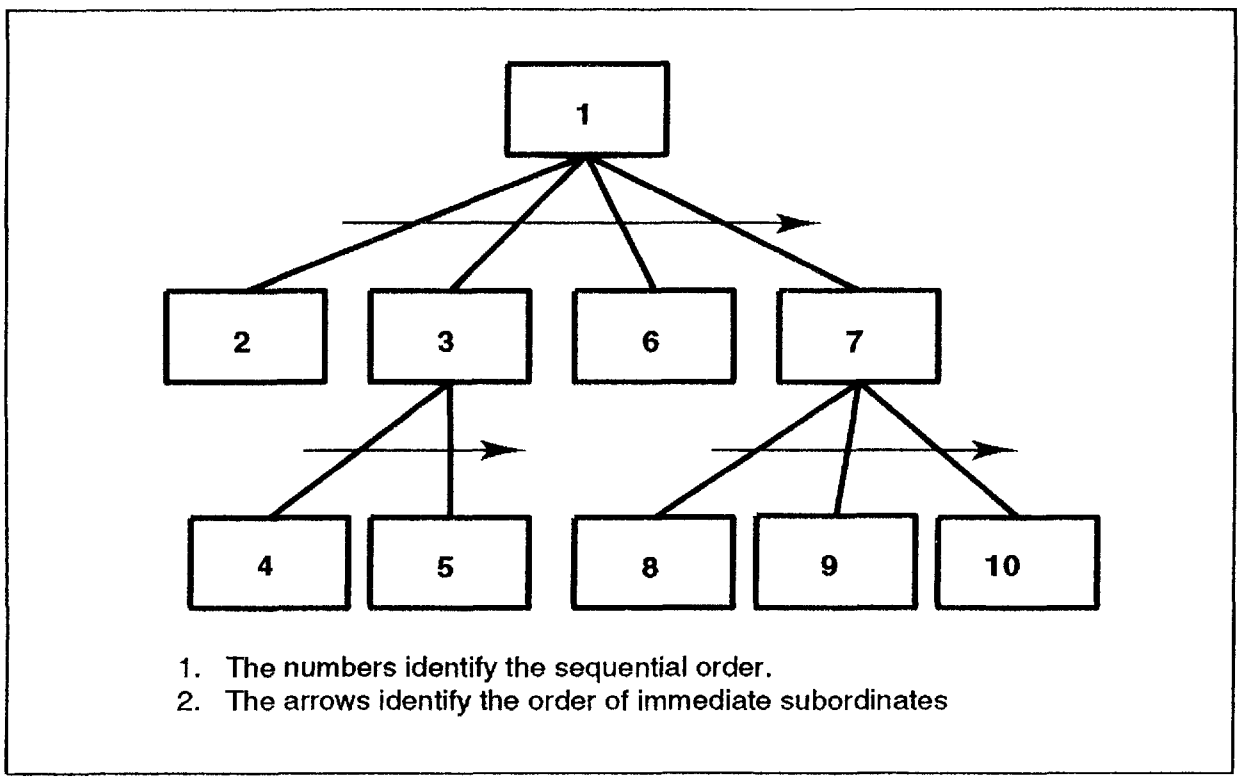

Figure 5 — Relationship of Hierarchical Structure in Sequential Order

5.3 Identification of Information Objects

Information objects in the environment of the Presentation Process which are used in the presentation of documents are identified by

— an ASN.1 object identifier, whose syntax is defined by ISO/IEC 8824;

— a public identifier, whose syntax is defined by ISO/IEC 9070;

— an Environment Name, whose syntax is defined by this International Standard.

Some of these information objects are defined by this International Standard, whereas others are defined elsewhere.

5.3.1 Public Object Identifiers

ASN.1 object identifiers and public identifiers defined by ISO/IEC 9070 are referred to as Public Object Identifiers.

NOTES

7 Public Object Identifiers provide unambiguous identification of information objects in open interchange, even in situations in which the information object identified may not be known.

8 An information object that is used in an ASN.1 environment will generally be assigned an ASN.1 object identifier value.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642506 258

9 An information object that is used in an SGML environment will generally be assigned an ISO/IEC 9070 public identifier value.

10 There is, in general, no conversion or mapping that can be used to convert an ASN.1 object identifier value into an ISO/IEC 9070 public identifier which identifies the same information object. Similarly, there is no way to convert an ISO/IEC 9070 public identifier into an ASN.1 object identifier value which identifies the same information object.

11 To allow both structure representation and interchange formats to carry both kinds of Public Object Identifier, the Binary Structure Representation and Interchange Format defines a representation for ISO/IEC 9070 public identifiers, and the Clear Text Structure Representation and Interchange Format defines a representation for ASN.1 object identifiers. These representations apply only to this International Standard.

5.3.2 Assignment of Public Object Identifiers

This International Standard defines Public Object Identifiers for the identification of several information objects. For each such information object, this International Standard specifies both

— an ASN.1 object identifier value as defined in ISO/IEC 8824;

— a public identifier as defined in ISO/IEC 9070,

which identify that information object.

Each information object for which this International Standard defines a Public Object Identifier is identified by an *Object Name*. This Object Name is the "object name" part of the ISO/IEC 9070 public identifier which is assigned to it. The full ISO/IEC 9070 public identifier and the ASN.1 object identifier value which identify the information object are specified in annex B

The identification of an information object which is defined by this International Standard in an instance of SPDL Structure should use:

— the public identifier value as defined by ISO/IEC 9070 if the Clear Text Structure Representation and Interchange Format is being used;

— the ASN.1 object identifier value as defined by ISO/IEC 8824 if the Binary Structure Representation and Interchange Format is being used.

5.3.3 Notation

The text of this International Standard describes the effect of several parameter values which are specified by means of a Public Object Identifiers In these cases, the parameter value represented by the Public Object Identifier is represented by the Object Name corresponding to that Public Object Identifier in italics, e.g. *dpi:plex:simplex*.

5.3.4 Environment Names

An information object in the environment of the Presentation Process may also be identified by an Environment Name. An Environment Name consists of single object-name component as defined by ISO/IEC 9070.

An Environment Name is semantically equivalent to the ISO/IEC 9070 public identifier for which the owner name is the word "UNREGISTERED" and the object name consists of a single object-name component. Specifically, the Environment Name "*objectname*" is semantically equivalent to the public identifier for which

— the owner name consists of the unregistered owner prefix "UNREGISTERED", with no owner name components;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— the object name consists of the single object-name component "*objectname*".

NOTE 12 Environment Names are meaningful only within a particular domain of use. This domain may consist of a single system, a set of related systems, or a broader domain in which a set of naming agreements has reached. Environment Names may be ambiguous if used in open interchange.

5.4 Representation and Interchange Format

The SPDL structure is easily parsable by a variety of processes. These include imposition processes and processes which extract pages or pictures from SPDL documents as well as Presentation Processes. To support the parsing of the SPDL structure by a variety of processes in a variety of environments, both ASN.1 and SGML encodings of the SPDL structure are standardized. An SPDL Presentation Process only needs to be able to process one of the representations to conform to this International Standard.

The SPDL structure enables processes to extract elements from SPDL documents together with relevant attributes, and to determine external resource requirements, without having to interpret the content elements which specify page and picture images. No process other than an SPDL Presentation Process needs to be able to process the content notation.

The SPDL structure specifies the logical structure of an instance of SPDL. Not every structure element need have an explicit representation in the Structure Representation and Interchange Format, so long as the logical structure of each instance of SPDL is unambiguously represented.

This International Standard specifies two representation and interchange formats for the encoding of the SPDL structure:

— a binary encoding called the Binary Structure Representation and Interchange Format;

— a clear text encoding called the Clear Text Structure Representation and Interchange Format.

The Binary Structure Representation and Interchange Format is based on ASN.1. The Clear Text Structure Representation and Interchange Format is based on SGML. The Binary Structure Representation and Interchange Format and the Clear Text Structure Representation and Interchange Format are fully equivalent: any functionality that can be expressed in one can be expressed in the other with a simple syntactic transformation.

The Structure Representation and Interchange Formats ensure that every instance of SPDL is self-identifying within the domain of use of the structure notations on which the Structure Representation and Interchange Formats are based.

This International Standard also specifies two Representation and Interchange Formats for the encoding of the token sequences which comprise the content:

— a compact binary encoding called the Binary Content Representation and Interchange Format;

— a human readable clear text encoding called the Clear Text Content Representation and Interchange Format.

For reasons of compactness and efficiency of interpretation, specialized representations optimized for compactness and ease of interpretation are used.

The Binary Content Representation and Interchange Format is used with the Binary Structure Representation and Interchange Format. The Clear Text Content Representation and Interchange Format is used with the Clear Text Structure Representation and Interchange Format.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5.5 Instances of the Standard Page Description Language

An instance of the Standard Page Description Language may represent a document, define or undefine a resource, or encapsulate a structure element which may be included in another instance of SPDL by reference.

An *SPDL Instance* consists of an SPDL Identifier and a TOP LEVEL STRUCTURE structure element. The SPDL Identifier identifies an SPDL Instance unambiguously as such among data structures which are encoded using the structure notation used by the Structure

Representation and Interchange Format (ASN.1 or SGML). The TOP LEVEL STRUCTURE structure element is the structure element at the highest structure level of the SPDL Instance.

5.5.1 Binary Structure Representation and Interchange Format

An SPDL Instance in the Binary Structure Representation and Interchange Format shall be encoded as an instance of the ASN.1 EXTERNAL type for which

— the *direct-reference* is the object identifier {iso(1) standard(0) 10180 external-id(0)};

— the *indirect-reference* is absent;

— the *data-value-descriptor* is absent;

— the *encoding* is a *single-ASN.1-type* value which consists of a TOP LEVEL STRUCTURE structure element in the Binary Structure Representation and Interchange Format.

5.5.2 Clear Text Structure Representation and Interchange Format

An SPDL Instance in the Clear Text Structure Representation and Interchange Format shall be encoded as an SGML document instance for which

— the SGML *Document Type Declaration* begins with the character string "<!DOCTYPE SPDL";

— the SGML document type declaration subset has only entity declarations in which the entity text consists of an SGML external entity specification;

— following the SGML document type declaration there is a single SGML document element whose content is the representation of a TOP LEVEL STRUCTURE structure element in the Clear Text Structure Representation and Interchange Format.

5.5.3 TOP LEVEL STRUCTURE

A TOP LEVEL STRUCTURE structure element shall be one of:

— a DOCUMENT structure element;

— an ENVIRONMENT RESOURCE structure element;

— an INCLUDABLE STRUCTURE structure element;

which is not subordinate to any other structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642509 T67

5.5.4 Comments

The Structure Representation and interchange Format allows for the the inclusion of comments in any instance of SPDL. The method of including comments in an instance of SPDL Instance depends on the Structure Representation and Interchange Format being used. Comments shall have no effect on either structure processing or content processing.

Comments consist of IA5Strings (as defined in ISO/IEC 8824) which do not interfere with the parsing of the chosen Structure Representation and Interchange Format.

NOTE 13 The set of comments which may occur depends on the Structure Representation and Interchange Format being used. This has no effect on the presentation of an SPDL document since comments are ignored by the Structure Processor.

6 Types

The system of types is a hierarchical system based on set inclusion.

— There is a set of *base types*. The sets of values of the base types are disjoint.

— There are types for which the set of values is a subset of the set of values of another type. If the set of values of type *A* is a subset of the set of values of type *B*, then type *A* is a *sub-type* of *B*.

— There are types for which the set of values includes the set of values of two or more other types. If the set of values of type *A* contains the set of values of type *B*, then type *A* is a *super-type* of *B*.

Sub-types of a given type are also sub-types of any super-type of the given type: that is, if *A* is a sub-type of *B* and *C* is a super-type of *B*, then *A* is a sub-type of *C*.

The value of each base structure element belongs to one the types defined in this clause. The definitions identify each type as a base type, a sub-type of a base type, or a super-type of two or more other types; and identifies the set of values of the type.

Equality for values of a type is defined for each of the base types for which a definition of equality is required. Two values of a type which is a sub-type of a single base type are equal if and only if they are equal as values of the base type. Equality for values of a type which includes values of more than one base type is defined for each such type.

NOTES

14 Two values of a type which is a super-type of two or more base types are always equal if they are values of the same base type and are equal as values of that base type.

15 With the exception of the Glyph Identifier type, two values of a type which is a super-type of two or more base types are equal only if they are values of the same base type and are equal as values of that base type. In the case of the of the Glyph Identifier type, two values which are of different subtypes may also be equal: see 6.2.

6.1 Boolean type

The *Boolean type* is a base type. The set of values of the *Boolean* type consists of two values, *true* and *false*. Two values of the *Boolean* type are equal if and only if they are the same.

6.2 Enumerated types

There are a number of *Enumerated types*. Each Enumerated type is a base type. The set of values of each Enumerated type consists of a finite list of values. The meaning of the values is specified by the type definition. The representation of the values depends on the Structure Representation and Interchange Format being used.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642510 789

Two Enumerated type values are equal if and only if they are values of the same Enumerated type and are the same.

NOTE 16 Values of two distinct Enumerated types are always unequal.

6.3 Integer type

The *Integer type* is a base type. The set of values of the *Integer* type consists of the subset of the mathematical integers that are greater than or equal to the integer *MinInteger* and less than or equal to the integer *MaxInteger*. Two values of the *Integer* type are equal if and only if they are numerically equal.

The values of the integers *MinInteger* and *MaxInteger* are system dependent. The minimum value for *MaxInteger* and the maximum value for *MinInteger* are specified in clause 39.

NOTE 17 The set of values of the Integer type includes all of the the integers between $-2^{31}+1$ and $2^{31}-1$ inclusive.

6.4 Cardinal type

The *Cardinal type* is a sub-type of the *Integer* type The set of values of the *Cardinal* type consists of the non-negative values of the *Integer* type.

6.5 Positive Integer Type

The *Positive Integer type* is a sub-type of the *Integer* type. The set of values of the *Positive Integer* type consists of the positive values of the *Integer* type (excluding zero).

6.6 Real type

The *Real type* is a base type. The set of values of the *Real* type is a subset of the mathematical rational numbers Two values of the *Real* type are equal if and only if they are numerically equal.

The set of rational numbers included in the set of values of the *Real* type is system dependent. The minimum set of rational numbers which shall be included in the set of values of the *Real* type is specified in clause 39.

NOTE 18 The set of values of the *Real* type includes all of the rational numbers which can be represented as basic single width floating point binary numbers as specified by IEEE 754:1987

6.7 Number type

The *Number type* is a super-type of the *Integer* type and the *Real* type. The set of values of the *Number* type consists of the union of the set of values of the *Integer* type and the set of values of the *Real* type. Two values of the *Number* type are equal if and only if they are both values of the same base type and are equal as values of that base type.

NOTES

19 A value of the *Real* type is distinct from a value of the *Integer* type even if the two values are numerically equivalent.

20 There are some circumstances under which values of *Integer* type are automatically converted to the numerically equivalent values of the *Real* type. For example, see 20.1.8 (**Equal** operator).

6.8 Non-Negative Number type

The *Non-Negative Number type* is a sub-type of the *Number* type. The set of values of the *Non-Negative Number* type consists of the union of the set of values of the *Cardinal* type and the set of values of the *Real* type which

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

are greater than zero. Two values of the *Non-Negative Number* type are equal if and only if they are both values of the same base type and are equal as values of that base type.

6.9 Printable String type

The *Printable String type* is a base type. The set of values of the *Printable String* type consists of the set of PrintableStrings defined in ISO/IEC 8824. There is no definition of equality for values of the *Printable String* type.

6.10 Octet String type

The *Octet String type* is a base type. The set of values of the *Octet String* type consists of all sequences of octets. There is no definition of equality for values of the *Octet String* type.

NOTE 21 Instances of the Content Representation and Interchange Format are represented in structure by values of the *OctetString* type.

6.11 Name type

The *Name type* is a sub-type of the *Identifier* type. The set of values of the *Name* type is defined in 18.1.8. Two values of the *Name* type are equal if the corresponding octet strings, as defined in 18.1.8, are the same.

6.12 Object Identifier type

The *Object Identifier type* is a base type. The set of values of the *Object Identifier* type, consists of the set of object identifier values defined by ISO/IEC 8824. Two values of the *Object Identifier* type are equal if and only if they consist of the same sequence of numbers.

The content representation of an Object Identifier is an *OctetStringReference* which references an *OctetString* which contains the character string representation of the Object Identifier value encoded using ISO/IEC 646 IRV. The character string representation of the Object Identifier value consists of a sequence of numeric strings S_1 . . S_n, separated by period (".") characters Each of the numeric strings consists of either the single digit "0" or one or more of the digits 0 . . 9, beginning with a digit other than "0". The first string S_1 is the decimal representation of the first number of the Object Identifier value, the second string S_2 is the decimal representation of the second number of the Object Identifier value, and so on, with the last string S_n representing the last number of the Object Identifier value.

6.13 Public Identifier type

The *Public Identifier type* is a base type The set of values of the *Public Identifier* type consists of the set of public identifiers as defined by ISO/IEC 9070. Two values of the *Public Identifier* type are equal if and only if they are equal as public identifiers

NOTE 22 ISO/IEC 9070 specifies that the canonical character string form of public identifiers be used for internal equality comparisons.

The content representation of a public identifier is an OctetStringReference which references an OctetString which contains the canonical character string form of the public identifier, encoded using ISO/IEC 646 IRV.

6.14 Public Object Identifier type

The *Public Object Identifier type* is a super-type of the *Object Identifier* type and the *Public Identifier* type. The set of values of the *Public Object Identifier* type consists of the union of the set of values of the *Object Identifier* type and the *Public Identifier* type. Two values of the *Public Object Identifier* type are equal if and only if they are both values of the same base type and are equal as values of that base type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642512 551

6.15 Environment Name type

The *Environment Name type* is a base type. The set of values of the *Environment Name* type consists of the set of "object-name components" as defined by ISO/IEC 9070. Two values of the *Environment Name* type are equal if and only if they consist of the same character string.

6.16 Environment Identifier type

The *Environment Identifier type* is a super-type of the *Object Identifier* type, the *Public Identifier* type, and the *Environment Name* type. The set of values of the *Environment Identifier* type consists of the union of the set of values of these types. Two values of the *Environment Identifier* type are equal if and only if they are both values of the same base type and are equal as values of that type.

6.16.1 Structured Glyph Name type

The *Structured Glyph Name type* is a base type. The set of values of the *Structured Glyph Name* type is the set of structured-names as defined by ISO/IEC 9541-1.

NOTE 23 The requirement that a structured-name be "an unambiguous identifier for some object suitable for representing that object within a decentralized, distributed computer system, invariant over space and time" requires that the structured-name have both an "Owner Name" and an "Object Name", and that the "Owner Name" identify the owner unambiguously.

Two values of the *Structured Glyph Name* type are equal if and only if the structured-names are equal.

NOTE 24 The public identifier which is semantically equivalent to the structured-name is defined by ISO/IEC 9070, which specifies that the canonical character string form of public identifiers be used for internal equality comparisons.

6.16.2 ISO10036 Glyph Name type

The *ISO10036 Glyph Name type* is a sub-type of the *Structured Glyph Name* type. The set of values of the *ISO10036 Glyph Name* type is the set of structured-names for which the Owner Name identifies the registration authority authorized by ISO/IEC 10036. Two values of the *ISO10036 Glyph Name* type are equal if and only if they are equal as Structured Glyph Names.

NOTE 25 The canonical character string form of the public identifier which is semantically equivalent to an *ISO10036 Glyph Name* is of the form "ISO/IEC 10036/RA//Glyphs::*nnnn*", where "*nnnn*" is a sequence of decimal digits beginning with a non-zero digit which represents an integer in the range of 1 to 2^{31}-1.

6.16.3 Simple Glyph Name type

The *Simple Glyph Name type* is a sub-type of the *Name* type. The set of values of the *Simple Glyph Name* type is the set of Names of length less than or equal to 100 characters which begin with a character other than the period (" ")

Values of the *Simple Glyph Name type* which are not AFII Glyph Names are semantically equivalent to public identifiers which have unregistered owner names. Specifically, the Simple Glyph Name *glyphname* is semantically equivalent to a public identifier for which

- the owner name consists of the unregistered owner prefix "UNREGISTERED", with no owner name components;
- the object name consists of the single object-name component *glyphname*.

NOTE 26 Two values of the Simple Glyph Name type are equal if and only if they are equal as Names.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.16.4 AFII Glyph Name type

The *AFII Glyph Name type* is a sub-type of the *Simple Glyph Name* type. The set of values of the *AFII Glyph Name* type is the set of Names for which the corresponding octet string, as defined in 18.1.8, is the encoding of a character string of the form "afii*nnnn*", where "*nnnn*" is a sequence of decimal digits beginning with a non-zero digit, which represents an integer in the range of 1 to 2^{31}-1.

The AFII Glyph Name "afii*nnnn*" is semantically equivalent to the structured-name for which the canonical character string form of the equivalent public identifier is "ISO/IEC 10036/RA//Glyphs::*nnnn*"

NOTE 27 Two values of the *AFII Glyph Name* type are equal if and only if they are equal as Names.

6.16.5 Glyph Identifier type

The *Glyph Identifier type* is a super-type of the *Structured Glyph Name* type and the *Simple Glyph Name* type. The set of values of the *Glyph Identifier* type consists of the union of the set of values of the *Structured Name* type and the set of values of the *Simple Glyph Name* type.

Two values of the Glyph Identifier type are equal if and only if one of the following is true:

— they are both values of the *Structured Glyph Name* type which are equal as Structured Glyph Names;

— they are both values of the *Simple Glyph Name* type which are equal as Names;

— one is a value of the *ISO10036 Glyph Name* type for which the canonical character string form of the semantically equivalent public identifier is "ISO/IEC 10036/RA//Glyphs::*nnnn*"; and the other is the value of the *AFII Glyph Name* type for which the corresponding octet string, as defined in 18 1.8, is the encoding of the character string "afii*nnnn*", where "*nnnn*" is the same sequence of decimal digits.

7 Document Structure

An SPDL document may represent either a single page or a set of pages. The primary structure elements from which an SPDL document is constructed are the PICTURE and PAGESET structure elements.

A PICTURE describes an image which can be considered as a single entity. It may be a page image or a partial page image. A PAGESET describes one or more pages.

There are two additional semantically distinct structure elements which are important to the SPDL document structure.

— A PAGE is a PICTURE which is not subordinate to any other PICTURE.

— A DOCUMENT is a PAGESET or a PICTURE which is not subordinate to any other structure element.

These structure elements are not syntactically distinct from the PICTURE and PAGESET structure elements, but are distinguished by their position in the hierarchical structure.

Each PAGESET or PICTURE is a BLOCK and has an associated Block State as defined in clause 16. Each PAGESET or PICTURE is either a DOCUMENT or the subordinate of a superior BLOCK.

7.1 DOCUMENT

A DOCUMENT structure element shall be one of:

— a PAGESET structure element;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642514 324

— a PAGE structure element;

which is not subordinate to any other structure element.

The processing of a DOCUMENT shall consist of:

a) resetting the mechanism for the presentation of the completed document;

b) initializing the presentation parameters;

c) if the DOCUMENT is a PAGESET, processing it as specified in 7.2;

d) if the DOCUMENT is a PAGE, processing it as specified in 7.3;

e) performing any finishing processes on the document;

f) presenting the completed document.

7.2 PAGESET

A PAGESET structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one PROLOGUE structure elements;

— a sequence of zero or more PAGESET or PAGE structure elements in any order.

The immediate subordinates shall occur in the above order

7.2.1 PAGESET Processing

The processing of a PAGESET shall consist of:

a) creating a Block State for the PAGESET as specified in 16.2;

b) processing the immediately subordinate PROLOGUE if present;

c) processing each of the immediately subordinate PAGESET and PAGE structure elements in their sequential order;

d) destroying the Block State for the PAGESET.

If the PAGESET has no subordinate PAGE structure elements, no page image shall be rendered as a result of processing the PAGESET.

7.3 PAGE

A PAGE structure element shall be a PICTURE which is not subordinate to any other PICTURE structure element.

The processing of a PAGE structure element shall consist of:

a) establishing a Current Page Image associated with the PAGE whose initial value is a blank page image as specified in 17.2;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

b) processing the PICTURE structure element as specified in 7.4;

c) rendering the Current Page Image associated with the PAGE on an instance of the medium.

The rendering of the Current Page Image associated with the PAGE on an instance of the medium is specified in clause 16.

7.4 PICTURE

A PICTURE structure element shall be a composite structure element which shall have the following immediate subordinates:

— one CONTENT REPRESENTATION IDENTIFIER structure element;

— one PICTURE BODY or NON-SPDL PICTURE BODY structure element.

The immediate subordinates shall occur in the above order.

A PICTURE which has an immediately subordinate PICTURE BODY shall be called an SPDL PICTURE. A PICTURE which has an immediately subordinate NON-SPDL PICTURE BODY shall be called a NON-SPDL PICTURE.

7.4.1 CONTENT REPRESENTATION IDENTIFIER

A CONTENT REPRESENTATION IDENTIFIER shall be a base structure element. The value of a CONTENT REPRESENTATION IDENTIFIER shall be of type *Public Object Identifier*.

The Object Names for which Public Object Identifiers are defined by this International Standard and the meaning of those values shall be as shown in table 1.

Table 1

Object Name	meaning
Content::SPDL-Binary	the PICTURE has an immediately subordinate PICTURE BODY for which the content of each immediately subordinate TOKENSEQUENCE is an instance of the SPDL content notation in the Binary Content Representation and Interchange Format;
Content::SPDL-ClearText	the PICTURE has an immediately subordinate PICTURE BODY for which the content of each immediately subordinate TOKENSEQUENCE is an instance of the SPDL content notation in the Clear Text Content Representation and Interchange Format;

Any other value of the CONTENT REPRESENTATION IDENTIFIER specifies that the PICTURE has an immediately subordinate NON-SPDL PICTURE BODY structure element and identifies the format of the NON-SPDL PICTURE BODY structure element.

7.4.2 PICTURE BODY

A PICTURE BODY structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one PROLOGUE structure elements;

— a sequence of zero or more PICTURE or TOKENSEQUENCE structure elements in any order.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642516 1T7

The immediate subordinates shall occur in the above order. The TOKENSEQUENCE structure element is defined in 16.5.

7.4.3 NON-SPDL PICTURE BODY

The NON-SPDL PICTURE BODY structure element is outside of the scope of this International Standard. The structure of a NON-SPDL PICTURE BODY is identified by the value of the CONTENT REPRESENTATION IDENTIFIER of the most immediately superior PICTURE, and is defined by owner of that Public Object Identifier.

7.4.4 SPDL PICTURE Processing

The processing of an SPDL PICTURE shall consist of:

a) creating a Block State for the PICTURE as specified in 16.2;

b) processing the immediately subordinate PROLOGUE if present;

c) processing each of the PICTURE or TOKENSEQUENCE structure elements which is immediately subordinate to the subordinate PICTURE BODY in their sequential order;

d) destroying the Block State for the PICTURE.

The processing of each TOKENSEQUENCE structure element which is immediately subordinate to the subordinate PICTURE BODY shall consist of

a) Calling the Content Processor with:

   — a content value which is the value of the TOKENSEQUENCE;

   — a Context of Interpretation which is obtained from the Block State of the most immediately superior PICTURE as specified in 16.4.2;

   — the Current Page Image of the most immediately superior PAGE;

b) Upon completion of content processing,

   — replacing the Current Virtual Machine State of the Block State of the most immediately superior PICTURE with the Virtual Machine State returned by the Content Processor;

   — replacing the Current Page Image of the most immediately superior PAGE with the Current Page Image returned by the Content Processor.

7.4.5 NON-SPDL PICTURE Processing

The processing of a NON-SPDL PICTURE depends on whether the Structure Processor is able to recognize the value of the CONTENT REPRESENTATION IDENTIFIER and to process the NON-SPDL PICTURE BODY as specified by that value. The processing of a NON-SPDL PICTURE shall consist of one of:

-- processing the NON-SPDL PICTURE BODY as specified by the value of the CONTENT REPRESENTATION IDENTIFIER;

— raising a structure error, ignoring the NON-SPDL PICTURE BODY, and continuing document processing, if permitted by the value of the *abort-policy* parameter.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642517 033

7.5 PROLOGUE

A PROLOGUE structure element and all of the subordinates of a PROLOGUE structure element shall be called *Prologue Structure Elements.* A PROLOGUE structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or more EXTERNAL DECLARATION structure elements;

— zero or one INFORMATIVE DECLARATION structure element;

— zero or more NON-SPDL OPERATION structure element;

— zero or more DPI DECLARATION structure elements;

— zero or one CONTEXT DECLARATION structure element;

— zero or more RESOURCE DEFINITION, RESOURCE DECLARATION, or CONTEXT ADDITION structure elements in any order;

— zero or one SETUP PROCEDURE structure elements.

The immediate subordinates shall occur in the above order.

The processing of a PROLOGUE structure element shall consist of

a) processing each of the immediately subordinate structure elements in their order of occurrence;

b) at the end of the processing of the immediately subordinate structure elements, setting the Inheritable Context Stack of the Block State of the most immediately superior BLOCK to a copy of the Context Stack of the Current Virtual Machine State of that Block State.

7.6 INFORMATIVE DECLARATION

An INFORMATIVE DECLARATION structure element provides information about the BLOCK in the form of hints. A *hint* is information which is not required to be accurate or complete, but which may still be useful. An example of a hint is a list of fonts or other resources required for the presentation of the document.

An INFORMATIVE DECLARATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or more HINT structure elements.

7.6.1 HINT

A HINT structure element shall be a composite structure element which shall have the following immediate subordinates:

— one HINT NAME structure element;

— one HINT VALUE structure element.

The immediate subordinates shall occur in the above order.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642518 T7T

7.6.2 HINT NAME

A HINT NAME structure element shall be a base structure element. The value of a HINT NAME structure element shall be of type *Environment Identifier*.

7.6.3 HINT VALUE

A HINT VALUE structure element may be a base structure element or a composite structure element, depending on the value of the peer HINT NAME.

If the HINT VALUE is a base structure element, the type of the value shall depend on the value of the peer HINT NAME. If the HINT VALUE is a composite structure element, the immediate subordinates shall depend on the value of the peer HINT NAME.

7.6.4 INFORMATIVE DECLARATION Processing

An INFORMATIVE DECLARATION structure element shall have no effect on document presentation.

7.7 Hints

The following HINT structure elements are defined by this International Standard.

7.7.1 Bounding Box

This HINT identifies the size of the smallest bounding box which will contain all of the images specified by the content of the PAGESET or PICTURE, in the units of the User Coordinate System in effect when the PAGESET or PICTURE is used

7.7.1.1 HINT NAME

The value of the subordinate HINT NAME structure element shall be the Public Object Identifier corresponding to the Object Name *Hints::BoundingBox*.

7.7.1.2 HINT VALUE

The subordinate HINT VALUE structure element shall be a composite structure element which shall have the following immediate subordinates:

— one MINIMUM X structure element;

— one MINIMUM Y structure element;

— one MAXIMUM X structure element;

— one MAXIMUM Y structure element

The immediate subordinates shall occur in the above order.

7.7.1.3 MINIMUM X

A MINIMUM X structure element shall be a base structure element. The value of a MINIMUM X structure element shall be of type *Number*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642519 906

7.7.1.4 MINIMUM Y

A MINIMUM Y structure element shall be a base structure element. The value of a MINIMUM Y structure element shall be of type *Number*.

7.7.1.5 MAXIMUM X

A MAXIMUM X structure element shall be a base structure element. The value of a MAXIMUM X structure element shall be of type *Number*.

7.7.1.6 MAXIMUM Y

A MAXIMUM Y structure element shall be a base structure element. The value of a MAXIMUM Y structure element shall be of type *Number*.

7.7.2 Number of Pages

This HINT identifies the number of PAGE structure elements in a PAGESET.

7.7.2.1 HINT NAME

The value of the subordinate HINT NAME shall be the Public Object Identifier corresponding to the Object Name *Hints::NumberOfPages*.

7.7.2.2 HINT VALUE

The subordinate HINT VALUE structure element shall be a base structure element. The value of the subordinate HINT VALUE structure element shall be of type *Cardinal*.

7.7.3 Orientation

This HINT identifies orientation of the image generated by a PICTURE or the page images generated by a PAGESET.

7.7.3.1 HINT NAME

The value of the subordinate HINT NAME structure element shall be the Public Object Identifier corresponding to the Object Name *Hints::Orientation*.

7.7.3.2 HINT VALUE

The subordinate HINT VALUE structure element shall be a base. structure element. The type of the value of the subordinate HINT VALUE structure element shall be an Enumerated type which identifies an orientation. The values of this type and the orientations identified shall be as shown in table 2.

Table 2

Value	Orientation
portrait	portrait (shorter dimension parallel to the x-axis);
landscape	landscape (longer dimension parallel to the x-axis);

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642520 628

7.7.4 Font References

This HINT identifies the Font References included in FONT REFERENCE FONT SPECIFICATION structure elements in the immediately superior BLOCK or in subordinate BLOCKs.

7.7.4.1 HINT NAME

The value of the subordinate HINT NAME shall be the Public Object Identifier corresponding to the Object Name *Hints::FontReferences*.

7.7.4.2 HINT VALUE

The subordinate HINT VALUE shall be a composite structure element which shall have the following immediate subordinates:

— one or more FONT REFERENCE structure elements.

7.7.5 External References

This HINT identifies the structure elements included by reference in the immediately superior BLOCK or in subordinate BLOCKs

7.7.5.1 HINT NAME

The value of the subordinate HINT NAME shall be the Public Object Identifier corresponding to the Object Name *Hints::ExternalReferences*.

7.7.5.2 HINT VALUE

The subordinate HINT VALUE shall be a composite structure element which shall have the following immediate subordinates:

— one or more EXTERNAL DECLARATION structure elements.

7.7.6 Resources Used

This HINT identifies the resources declared in RESOURCE DECLARATION or RESOURCE DEFINITION structure elements in the immediately superior BLOCK or in subordinate BLOCKs.

7.7.6.1 HINT NAME

The value of the subordinate HINT NAME shall be the Public Object Identifier corresponding to the Object Name *Hints::ResourcesUsed*

7.7.6.2 HINT VALUE

The subordinate HINT VALUE shall be a composite structure element which shall have the following immediate subordinates:

— one or more RESOURCE DECLARATION structure elements.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642521 564

7.7.7 Color Space Families Used

This HINT identifies the color space families of color space resources used or COLOR SPACE SPECIFICATIONs in the immediately superior BLOCK or in subordinate BLOCKs.

7.7.7.1 HINT NAME

The value of the subordinate HINT NAME shall be the Public Object Identifier corresponding to the Object Name *Hints::ColorSpaceFamiliesUsed*.

7.7.7.2 HINT VALUE

The subordinate HINT VALUE shall be a composite structure element which shall have the following immediate subordinates:

— one or more COLOR SPACE FAMILY structure elements.

7.7.8 Media Used

This HINT identifies the media declared in MEDIUM DECLARATION structure elements in the immediately superior BLOCK or in subordinate BLOCKs.

7.7.8.1 HINT NAME

The value of the subordinate HINT NAME shall be the Public Object Identifier corresponding to the Object Name *Hints::MediaUsed*.

7.7.8.2 HINT VALUE

The subordinate HINT VALUE shall be a composite structure element which shall have the following immediate subordinates:

— one or more MEDIUM DECLARATION structure elements.

7.7.9 Colorant Sets Used

This HINT identifies the colorant sets identified in COLORANT SET DPI structure elements in the immediately superior BLOCK or in subordinate BLOCKs.

7.7.9.1 HINT NAME

The value of the subordinate HINT NAME shall be the Public Object Identifier corresponding to the Object Name *Hints::ColorantSetsUsed*.

7.7.9.2 HINT VALUE

The subordinate HINT VALUE shall be a composite structure element which shall have the following immediate subordinates:

— one or more COLORANT SET SPECIFICATION structure elements.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642522 4T0

7.8 NON-SPDL OPERATION

A NON-SPDL OPERATION structure element may be used to specify an operation which may be performed by processes other than SPDL Presentation Processes which process or present SPDL documents. Operations which may be specified by NON-SPDL OPERATION structure elements are outside of the scope of this International Standard. A NON-SPDL OPERATION structure element has no effect on document presentation by an SPDL Presentation Process.

A NON-SPDL OPERATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one OPERATION NAME structure element;

— one OPERATION VALUE structure element.

The immediate subordinates shall occur in the above order.

7.8.1 OPERATION NAME

An OPERATION NAME structure element shall be a base structure element. The value of a OPERATION NAME shall be of type *Environment Identifier*.

7.8.2 OPERATION VALUE

An OPERATION VALUE structure element may be a base structure element or a composite structure element, depending on the value of the peer OPERATION NAME If the OPERATION VALUE is a base structure element, the type of the value shall depend on the value of the peer OPERATION NAME. If the OPERATION VALUE is a composite structure element, the immediate subordinates shall depend on the value of the peer OPERATION NAME.

7.8.3 NON-SPDL OPERATION Processing

The specification of the semantics of an NON-SPDL OPERATION and of the syntax of the OPERATION VALUE corresponding to any particular OPERATION NAME value shall be the responsibility of the owner of the Public Object Identifier which is the value of the OPERATION NAME.

A NON-SPDL OPERATION structure element shall have no effect on document presentation by an SPDL Presentation Process The effect of NON-SPDL OPERATION on the presentation of SPDL documents by processes other than an SPDL Presentation Process is outside of the scope of this International Standard.

NOTE 28 NON-SPDL OPERATION structure elements may affect the presentation of SPDL documents by presentation processes other than SPDL Presentation Processes.

7.9 CONTEXT DECLARATION

A CONTEXT DECLARATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or more INTERNAL RESOURCE IDENTIFIER structure elements.

The INTERNAL RESOURCE IDENTIFIER structure element is defined in 9.4.1.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.9.1 CONTEXT DECLARATION Processing

If the CONTEXT DECLARATION consists of *n* INTERNAL RESOURCE IDENTIFIER structure elements whose values are $name_1 \ldots . name_n$, if for each *i*, $1 \sigma i \sigma n$,

— $name_i$ has been bound to the Context Dictionary $Dict_i$ in the context of the CONTEXT DECLARATION;

— $DictRef_i$ is a *DictionaryReference* which references the Context Dictionary $Dict_i$;

and if *SystemDictRef* and *UserDictRef* are *DictionaryReferences* which reference the Context Dictionaries defined in 17.1.6, then the processing of the CONTEXT DECLARATION shall consist of replacing the Context Stack of the Block State of the most immediately superior BLOCK with the Context Stack consisting of (using the notation of 17.4.8):

<$DictRef_n$: *DictionaryReference*>

. . .

<$DictRef_1$: *DictionaryReference*>

<*UserDictRef*: *DictionaryReference*>

<*SystemDictRef*: *DictionaryReference*>

A structure error shall occur if any of the values $name_1 \ldots . name_n$ has not been bound to a Context Dictionary in the context of the CONTEXT DECLARATION.

If the CONTEXT DECLARATION has no subordinate INTERNAL RESOURCE IDENTIFIER structure elements, the processing of the CONTEXT DECLARATION shall consist of replacing the Context Stack of the Block State of the most immediately superior BLOCK with the Context Stack consisting of:

<*UserDictRef*: *DictionaryReference*>

<*SystemDictRef*: *DictionaryReference*>

7.10 CONTEXT ADDITION

A CONTEXT ADDITION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one INTERNAL RESOURCE IDENTIFIER structure element.

The INTERNAL RESOURCE IDENTIFIER structure element is defined in 9.4.1.

7.10.1 CONTEXT ADDITION Processing

The processing of a CONTEXT ADDITION shall consist of pushing the *DictionaryReference* which is bound to the value of the subordinate INTERNAL RESOURCE IDENTIFIER on the Context Stack of the Block State of the most immediately superior BLOCK.

A structure error shall occur if the value of the INTERNAL RESOURCE IDENTIFIER has not been bound to a Context Dictionary in the context of the CONTEXT ADDITION.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642524 273

7.11 SETUP PROCEDURE

A SETUP PROCEDURE shall be a composite structure element which shall have the following immediate subordinates:

— one or more TOKENSEQUENCE structure elements.

7.11.1 SETUP PROCEDURE Processing

The processing of a SETUP PROCEDURE shall consist of:

a) Calling the Content Processor with:

— a content value which is obtained by concatenating the values of the TOKENSEQUENCE structure elements which are immediately subordinate to the SETUP PROCEDURE;

— a Context of Interpretation which is obtained from the Block State of the most immediately superior BLOCK as specified in 16.4.2.

b) Upon completion of content processing, replacing the

— Graphics State Stack;

— Current Graphics State;

— ordered set of Save Objects;

— set of referenced objects;

— set of state variables;

of the Block State of the most immediately superior BLOCK with the corresponding elements of the Virtual Machine State returned by the Content Processor. A structure error or warning shall occur if the status returned by the Content Processor is other than *OK*.

No changes are made to the Context Stack or Operand Stack of the Block State of the most immediately superior BLOCK or to the Current Page Image as a result of the processing of a SETUP PROCEDURE.

NOTE 29 A SETUP PROCEDURE in the PROLOGUE of a PAGESET can set up the Current Graphics State, the set of referenced objects, and the set of state variables of the Initial Block State for TOKENSEQUENCE and PAGE structure elements immediately subordinate to that PAGESET.

8 External Structure Elements

An *External Structure Element* is a structure element which is available in the environment of the Presentation Process External Structure Elements may be included by reference within the structure of an SPDL Instance. An External Structure Element may be a structure element defined by this International Standard or a data structure defined elsewhere (such as a Font Attribute Set as defined by ISO/IEC 9541) which is a valid element of the structure of an SPDL Instance.

An SPDL Instance that includes an External Structure Element by reference must declare it in an EXTERNAL DECLARATION. The EXTERNAL DECLARATION identifies a data element in the environment of the Presentation Process and binds an Identifier to the External Structure Element contained in that data element. The way in which the External Structure Element is obtained from the referenced data element depends on the referenced data element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

An External Structure Element which is declared in an EXTERNAL DECLARATION may be included by reference in an SPDL Instance by placing an EXTERNAL REFERENCE in the SPDL Instance. The EXTERNAL REFERENCE can be placed at any position in the structure of SPDL Instance the within the scope of the binding of the Identifier to the External Structure Element where the External Structure Element could occur. The Structure Processor processes the EXTERNAL REFERENCE structure element by substituting the referenced External Structure Element for the EXTERNAL REFERENCE structure element.

The definitions of structure elements and their subordinates within this International Standard take no further note of the fact that an EXTERNAL REFERENCE may be substituted for any of the structure elements specified.

8.1 INCLUDABLE STRUCTURE

An INCLUDABLE STRUCTURE structure element is a structure element which is defined by this International Standard and which can be included in another SPDL Instance by the use of an EXTERNAL REFERENCE structure element. An INCLUDABLE STRUCTURE structure element shall be one of:

— a PAGESET structure element;

— a PICTURE structure element;

— a PICTURE BODY structure element;

— a NON-SPDL PICTURE BODY structure element;

— a PROLOGUE structure element;

— a INFORMATIVE DECLARATION structure element;

— a HINT structure element;

— a CONTEXT DECLARATION structure element;

— a RESOURCE DEFINITION structure element;

— a SETUP PROCEDURE structure element;

— a TOKENSEQUENCE structure element;

— a FONT REFERENCE structure element;

— a DICTIONARY RESOURCE SPECIFICATION structure element;

— a FONT SPECIFICATION structure element;

— a FONT REFERENCE FONT SPECIFICATION structure element;

— a FONTTYPE1 SPECIFICATION structure element;

— a REMAPPED FONT SPECIFICATION structure element;

— a COMPOSITE FONT SPECIFICATION structure element;

— a USER FONT SPECIFICATION structure element;

— a FONT PROPERTIES structure element;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642526 046

— a GLYPH INDEX MAP SPECIFICATION structure element;

— a GLYPH INDEX TRANSLATION TABLE structure element;

— a FONT INDEX MAP structure element;

— a DATA SOURCE SPECIFICATION structure element;

— a COLOR SPACE RESOURCE SPECIFICATION structure element;

— a PATTERN RESOURCE SPECIFICATION structure element;

— a FORM RESOURCE SPECIFICATION structure element;

— a DPI DECLARATION structure element;

— an ADDITIONAL DPI structure element.

8.2 EXTERNAL DECLARATION

An EXTERNAL DECLARATION structure element binds an Identifier to an External Structure Element. An EXTERNAL DECLARATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one STRUCTURE ELEMENT CLASS IDENTIFIER structure element;

— one STRUCTURE ELEMENT IDENTIFIER structure element;

— one LOCATION IDENTIFIER structure element

The immediate subordinates shall occur in the above order.

8.2.1 STRUCTURE ELEMENT CLASS IDENTIFIER

A STRUCTURE ELEMENT CLASS IDENTIFIER structure element shall be a base structure element. The type of the value of a STRUCTURE ELEMENT CLASS IDENTIFIER shall be an Enumerated type which identifies the structure element referenced by an EXTERNAL DECLARATION as an element of the specified class. The values of this type and the class of structure elements identified shall be as shown in table 3.

Table 3

Value	Class of Structure Elements
pageset	PAGESET structure elements
picture	PICTURE structure elements
picture-body	PICTURE BODY structure elements
non-spdl-picture-body	NON-SPDL PICTURE BODY structure elements
prologue	PROLOGUE structure elements
informative-declaration	INFORMATIVE DECLARATION structure elements
hint	HINT structure elements

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642527 T82

Value	Class of Structure Elements
context-declaration	CONTEXT DECLARATION structure elements
resource-definition	RESOURCE DEFINITION structure elements
dictionary-spec	DICTIONARY RESOURCE SPECIFICATION structure elements
setup-procedure	SETUP PROCEDURE structure elements
tokensequence	TOKENSEQUENCE structure elements
font-reference	FONT REFERENCE structure elements
font-attribute-set	Font Attribute Sets as defined by ISO/IEC 9541
font-properties	FONT PROPERTIES structure elements
font-spec	FONT SPECIFICATION, FONT IDENTIFIER FONT SPECIFICATION, FONT REFERENCE FONT SPECIFICATION, FONTTYPE1 FONT SPECIFICATION, REMAPPED FONT SPECIFICATION, COMPOSITE FONT SPECIFICATION, and USER FONT SPECIFICATION structure elements
font-ref-font-spec	FONT REFERENCE FONT SPECIFICATION structure elements
fonttype1-font-spec	FONTTYPE1 FONT SPECIFICATION structure elements
remapped-font-spec	REMAPPED FONT SPECIFICATION structure elements
composite-font-spec	COMPOSITE FONT SPECIFICATION structure elements
user-font-spec	USER FONT SPECIFICATION structure elements
glyph-index-map-spec	GLYPH INDEX MAP SPECIFICATION structure elements
gitt	GLYPH INDEX TRANSLATION TABLE structure elements
font-index-map	FONT INDEX MAPstructure elements
color-space-spec	COLOR SPACE RESOURCE SPECIFICATION structure elements
data-source-spec	DATA SOURCE SPECIFICATION structure elements
pattern-spec	PATTERN RESOURCE SPECIFICATION structure elements
form-spec	FORM RESOURCE SPECIFICATION structure elements
dpi-declaration	DPI DECLARATION structure elements
additional-dpi	ADDITIONAL DPI structure elements.

8.2.2 STRUCTURE ELEMENT IDENTIFIER

A STRUCTURE ELEMENT IDENTIFIER structure element shall be a base structure element. The value of a STRUCTURE ELEMENT IDENTIFIER shall be of type *Name*.

8.2.3 EXTERNAL DECLARATION Processing

Processing an EXTERNAL DECLARATION structure element shall consist of binding the value of the STRUCTURE ELEMENT IDENTIFIER to an External Structure Element. The scope of the binding shall be the set of all structure elements in the most immediately superior BLOCK which follow the EXTERNAL DECLARATION in the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

sequential order and all of their subordinates. The effect of the binding can be overridden by a subsequent EXTERNAL DECLARATION of the same kind in the same BLOCK or in a subordinate BLOCK.

If an EXTERNAL DECLARATION falls within the scope of a preceding EXTERNAL DECLARATION for which the value of the STRUCTURE ELEMENT IDENTIFIER is the same, the binding specified by the later EXTERNAL DECLARATION shall take precedence within its own scope.

The occurrence of an EXTERNAL DECLARATION for which either

— no External Structure Element which is identified by the value of the LOCATION IDENTIFIER is available in the environment of the Presentation Process;

— the syntax of the value of the LOCATION IDENTIFIER is not understood by the Structure Processor;

does not result in a structure error or warning.

NOTE 30 A structure error occurs only if the STRUCTURE ELEMENT IDENTIFIER is used in an EXTERNAL REFERENCE to reference the External Structure Element.

8.3 LOCATION IDENTIFIER

A LOCATION IDENTIFIER structure element identifies a data element which is either

— an External Structure Element;

— a data element to be used to generate a DATA SOURCE resource

When used to identify an External Structure Element, the data element identified by the LOCATION IDENTIFIER may be the External Structure Element itself, or it may be a data element containing the External Structure Element or a process which can be used to obtain the External Structure Element. The relationship between the identified data element and the referenced External Structure Element is specified in 8.4.

Since the methods by which information objects are identified differ between environments in which the encoding method used is predominantly ASN.1 and those in which the encoding method used is predominantly SGML, the structure of the LOCATION IDENTIFIER depends on the structure representation used. A LOCATION IDENTIFIER structure element shall be one of:

— a PUBLIC LOCATION IDENTIFIER structure element;

— a LOCAL LOCATION IDENTIFIER structure element;

— an ENCODING DEPENDENT LOCATION IDENTIFIER structure element.

8.3.1 PUBLIC LOCATION IDENTIFIER

A PUBLIC LOCATION IDENTIFIER structure element shall be a base structure element. The value of a PUBLIC LOCATION IDENTIFIER shall be of type *Public Object Identifier*.

The way in which the Public Object Identifier which is the value of the PUBLIC LOCATION IDENTIFIER identifies a data element in the environment of the Presentation Process is outside of the scope of this International Standard.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642529 855

8.3.2 LOCAL LOCATION IDENTIFIER

A LOCAL LOCATION IDENTIFIER structure element shall be a base structure element. The value of a LOCAL LOCATION IDENTIFIER shall be of type *Octet String*.

This value of a LOCAL LOCATION IDENTIFIER structure element identifies a data element in the local syntax. The way in which this value identifies a data element is outside of the scope of this International Standard.

NOTE 31 The value of a LOCAL LOCATION IDENTIFIER will commonly represent a file name in the local syntax. Since this value consists of private data, some substitution of data may be needed in open interchange. A comment may assist processes or human users in performing this substitution.

8.3.3 ENCODING DEPENDENT LOCATION IDENTIFIER

The ENCODING DEPENDENT LOCATION IDENTIFIER structure element depends on the structure representation used.

When the Binary Structure Representation and Interchange Format is used, an ENCODING DEPENDENT LOCATION IDENTIFIER shall be one of:

— a DOR LOCATION IDENTIFIER structure element;

— an IDENTIFIED SYNTAX LOCATION IDENTIFIER structure element.

When the Clear Text Structure Representation and Interchange Format is used, an ENCODING DEPENDENT LOCATION IDENTIFIER shall be one of:

— an SGML SYSTEM IDENTIFIER structure element;

— an SGML ENTITY structure element.

The way in which the ENCODING DEPENDENT LOCATION IDENTIFIER identifies a data element in the environment of the Presentation Process is outside of the scope of this International Standard.

8.3.3.1 DOR LOCATION IDENTIFIER

A DOR LOCATION IDENTIFIER structure element shall be a Distinguished Object Reference (DOR) as defined by ISO/IEC 10031-2.

8.3.3.2 IDENTIFIED SYNTAX LOCATION IDENTIFIER

An IDENTIFIED SYNTAX LOCATION IDENTIFIER structure element shall be an instance of the ASN.1 EXTERNAL type which shall have the following components:

— one "direct-reference" of type OBJECT IDENTIFIER;

— one "encoding" which is an instance of "single-ASN.1-type" of type ANY.

8.3.3.3 SGML SYSTEM IDENTIFIER

An SGML SYSTEM IDENTIFIER shall be an SGML external identifier as defined by ISO 8879.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642530 577

8.3.3.4 SGML ENTITY

An SGML ENTITY shall be an SGML general entity name as defined by ISO 8879.

NOTE 32 The location specified by this general entity name is given in an SGML entity declaration as part of the SGML document type declaration subset for the document (or SUBDOC) in which the general entity name is used.

8.4 Referenced Structure Element Identification

The value of the LOCATION IDENTIFIER may identify the External Structure Element directly, or it may identify a data element containing the External Structure Element or a process which can be used to obtain the External Structure Element. The identification of the External Structure Element which is identified by an EXTERNAL DECLARATION can be considered as a two step process:

— the LOCATION IDENTIFIER is used to identify the data element in the environment of the Presentation Process which contains (or generates) the referenced External Structure Element;

— the referenced External Structure Element is obtained from the data element identified in the first step.

If the value of the LOCATION IDENTIFIER identifies the External Structure Element directly, the data element identified in the first step is simply the External Structure Element itself

NOTES

33 An example of a data structure which contains a structure element would be a file in the local file system. Such a file might contain an SPDL Instance or a data structure containing one or more structure elements.

34 An example of a process which generates a structure element would be a relational data base program which can obtain a structure element from a data base

The data element which is identified by the LOCATION IDENTIFIER may be:

a) a data element consisting of an INCLUDABLE STRUCTURE structure element in the SPDL Representation and Interchange Format;

b) a data element consisting of an SPDL Instance in the SPDL Representation and Interchange Format;

c) a data element consisting of an INCLUDABLE STRUCTURE structure element in a non-standard format;

d) a process or an intelligent object which can generate an INCLUDABLE STRUCTURE structure element in either the SPDL Representation and Interchange Format or in a non-standard format.

The External Structure Element referenced in each of these cases shall be:

a) the structure element represented by the identified data element (in the SPDL Representation and Interchange Format);

b) the TOP LEVEL STRUCTURE structure element contained in the identified SPDL Instance in the identified data element (in the SPDL Representation and Interchange Format);

c) the structure element represented by the identified data element (in a format other than the SPDL Representation and Interchange Format, which must processable by the Presentation Process);

d) the structure element generated by the process or intelligent object which is identified by the LOCATION IDENTIFIER.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

NOTE 35 In case b, the process of obtaining the External Structure Element consists of removing the SPDL identification information from the SPDL Instance. The TOP LEVEL STRUCTURE may then be lexically included in the referencing document.

8.5 EXTERNAL REFERENCE

An EXTERNAL REFERENCE structure element shall be a base structure element. The value of an EXTERNAL REFERENCE shall be of type *Name*.

8.5.1 EXTERNAL REFERENCE Processing

Processing an EXTERNAL REFERENCE structure element shall consist of:

a) identifying the most immediately superior EXTERNAL DECLARATION which binds the value of the subordinate STRUCTURE ELEMENT IDENTIFIER to an External Structure Element;

b) obtaining the External Structure Element which is identified by the value of the LOCATION IDENTIFIER subordinate to the EXTERNAL DECLARATION identified in step a, as specified in 8.4;

c) verifying that the External Structure Element obtained in step b belongs to the class specified by the STRUCTURE ELEMENT CLASS IDENTIFIER subordinate to the EXTERNAL DECLARATION identified in step a;

d) if the structure element obtained in step b is in the SPDL Representation and Interchange Format, (lexically) replacing the EXTERNAL REFERENCE structure element with the External Structure Element and processing it as specified by this International Standard;

e) if the structure element obtained in step b is not in the SPDL Representation and Interchange Format, processing it as though the EXTERNAL REFERENCE had been replaced by an equivalent External Structure Element in the SPDL Representation and Interchange Format.

If any of the above steps fails the Structure Processor shall raise an error.

9 Resource Definitions

A *resource* is an information object, of a particular class, which is available in the environment of the Presentation Process. The following classes of resources are defined or used by this International Standard:

— Indexed Font

— Glyph Index Map

— Color Space

— Data Source

— Filter

— Pattern

— Form

— Context Dictionary

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642532 34T

For each class of resources, this International Standard specifies the way in which the use of the resource is invoked and the effect of the use of a resource in the processing of a document. The structure elements which are used for defining and accessing resources and the processing of those structure elements are specified in clause 9 and clause 10. The semantics of the resources are defined in the clauses which specify content processing.

9.1 General

Resources may be defined as specified in this International Standard or may be made available in the environment of the Presentation Process by means of a process outside of the scope of this International Standard.

9.1.1 Resource Definition

Resources which are defined as specified in this International Standard are defined by RESOURCE DEFINITION structure elements. A RESOURCE DEFINITION structure element may be subordinate to either:

— an ENVIRONMENT RESOURCE structure element;

— a DOCUMENT structure element.

A RESOURCE DEFINITION which is subordinate to an ENVIRONMENT RESOURCE defines a resource for inclusion in the environment of the Presentation Process. The scope of such a RESOURCE DEFINITION is system dependent.

A RESOURCE DEFINITION which is subordinate to a DOCUMENT defines a resource for use in the processing of the document. The scope of such a RESOURCE DEFINITION is the set of all structure elements in the most immediately superior BLOCK which follow the RESOURCE DEFINITION in the sequential order and all of their subordinates.

9.1.2 Resource Identification

A resource in the environment of the Presentation Process is identified by a combination of:

— a resource class;

— an Environment Identifier.

The Environment Identifier need only be unique within a particular resource class to unambiguously identify a resource.

A RESOURCE DEFINITION which is subordinate to an ENVIRONMENT RESOURCE structure element binds an Environment Identifier to the defined resource in the environment of the Presentation Process. The scope of this binding is system dependent.

A Document RESOURCE DEFINITION binds an Environment Identifier to the defined resource for the processing of the DOCUMENT. The scope of this binding is the set of all structure elements in the most immediately superior BLOCK which follow the RESOURCE DEFINITION in the sequential order.

9.1.3 Resource Undefinition

A resource in the environment of the Presentation Process can be undefined by a RESOURCE UNDEFINITION structure element. The effect of undefining a resource is to unbind the identifying Environment Identifier from the resource so that it can no longer be referenced by that Environment Identifier.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642533 286

9.1.4 Resource Reference Objects

With the exception of Context Dictionaries, resources which are used in content processing are represented by reference objects. The type of reference object used to represent a resource depends on the resource class shall be as shown in table 4.

Table 4

Resource	Reference Object
Context Dictionary	DictionaryReference
Indexed Font	DictionaryReference
Glyph Index Map	VectorReference
Color Space	VectorReference
Data Source	StreamObject
Filter	OctetStringReference
Pattern	DictionaryReference
Form	DictionaryReference

9.1.5 Resource Declaration

A document that uses a resource must declare it in a RESOURCE DECLARATION, which makes the resource available to the Content Processor as part of the Context of Interpretation.

A RESOURCE DECLARATION binds a Name to a reference object which references a resource. With the exception of Context Dictionaries, a resource which has been declared by a RESOURCE DECLARATION is made available for content processing by the execution of either

— a **FindResource** operator whose arguments are the resource class identifier and the Name which were bound to the resource by the RESOURCE DECLARATION;

— a **FindFont** operator whose argument is the Name which was bound to the Indexed Font resource by a RESOURCE DECLARATION.

Context Dictionaries, in contrast, are made available by means of CONTEXT DECLARATION or CONTEXT ADDITION structure elements which place them on the Context Stack.

The scope of a RESOURCE DECLARATION, and the scope of the binding of a Name to a reference object by a RESOURCE DECLARATION, consist of the set of all structure elements in the most immediately superior BLOCK which follow the RESOURCE DEFINITION in the sequential order, and all of their subordinates. The effect of the binding of a Name to a reference object by a RESOURCE DECLARATION can be overridden by subsequent RESOURCE DECLARATION in the same BLOCK or in a subordinate BLOCK.

9.2 ENVIRONMENT RESOURCE

An ENVIRONMENT RESOURCE structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or more EXTERNAL DECLARATION structure elements;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— zero or one INFORMATIVE DECLARATION structure element;

— zero or more RESOURCE DECLARATION or CONTEXT ADDITION structure elements in any order;

— one RESOURCE DEFINITION or RESOURCE UNDEFINITION structure element.

The immediate subordinates shall occur in the above order.

9.2.1 RESOURCE UNDEFINITION

A RESOURCE UNDEFINITION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one RESOURCE CLASS IDENTIFIER structure element;

— one ENVIRONMENT RESOURCE IDENTIFIER structure element.

The immediate subordinates shall occur in the above order.

9.2.2 ENVIRONMENT RESOURCE Processing

The processing of an ENVIRONMENT RESOURCE structure element depends on system dependent administrative policies and system resource availability. It may consist of either:

— processing the ENVIRONMENT RESOURCE structure element as described in the following paragraphs;

— ignoring the ENVIRONMENT RESOURCE structure element.

The administrative policies and system resource availability, and hence the effect of ENVIRONMENT RESOURCE processing, may differ from ENVIRONMENT RESOURCE to ENVIRONMENT RESOURCE and from time to time.

9.2.2.1 Subordinate RESOURCE DEFINITION

The processing of an ENVIRONMENT RESOURCE structure element which has a subordinate RESOURCE DEFINITION and which is not ignored shall consist of:

a) creating a Block State for the ENVIRONMENT RESOURCE structure element;

b) processing the subordinate EXTERNAL DECLARATION structure elements if present;

c) processing the subordinate RESOURCE DECLARATION or CONTEXT ADDITION structure elements if present;

d) processing the subordinate RESOURCE DEFINITION structure element;

e) destroying the Block State created in the step a.

A structure error shall occur if the processing of the subordinate RESOURCE DEFINITION results in a structure error or a structure warning.

9.2.2.2 Subordinate RESOURCE UNDEFINITION

The processing of a ENVIRONMENT RESOURCE structure element which has a subordinate RESOURCE UNDEFINITION and which is not ignored shall consist of deleting the binding of the Environment Identifier which

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642535 059

is the value of the subordinate ENVIRONMENT RESOURCE IDENTIFIER to any resource in the environment of the Presentation Process to which is:

— of the resource class specified by the subordinate RESOURCE CLASS IDENTIFIER;

— identified by the value of the subordinate ENVIRONMENT RESOURCE IDENTIFIER.

A structure warning shall occur if there is no resource in the environment of the Presentation Process which satisfies these conditions.

Whether or not the resource is actually deleted from the environment of the Presentation Process is system dependent. If the resource is identified by more than one Environment Identifier, the system may delete the resource itself (deleting the binding of any other Environment Identifiers to the resource), or delete only the binding of the current Environment Identifier to the resource (preserving the resource itself and other Environment Identifier bindings).

9.3 RESOURCE DEFINITION

A RESOURCE DEFINITION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one RESOURCE CLASS IDENTIFIER structure element;

— one ENVIRONMENT RESOURCE IDENTIFIER structure element;

— one RESOURCE SPECIFICATION structure element.

The immediate subordinates shall occur in the above order.

9.3.1 RESOURCE CLASS IDENTIFIER

A RESOURCE CLASS IDENTIFIER structure element shall be a base structure element. The type of the value of the RESOURCE CLASS IDENTIFIER shall be an Enumerated type. The values of this type and the class of resources identified shall be as shown in table 5.

Table 5

Value	Class of Resources
Dictionary	Context Dictionary
IndexedFont	Indexed Font
GlyphIndexMap	Glyph Index Map
ColorSpace	Color Space
DataSource	Data Source
Filter	Filter
Pattern	Pattern
Form	Form

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9.3.2 ENVIRONMENT RESOURCE IDENTIFIER

An ENVIRONMENT RESOURCE IDENTIFIER structure element shall be a base structure element. The value of an ENVIRONMENT RESOURCE IDENTIFIER shall be of type *Environment Identifier*.

The value of an ENVIRONMENT RESOURCE IDENTIFIER which is subordinate to an ENVIRONMENT RESOURCE should be an Object Identifier or a registered Public Identifier.

9.3.3 RESOURCE SPECIFICATION

A RESOURCE SPECIFICATION structure element shall be one of:

— a DICTIONARY SPECIFICATION structure element;

— a FONT SPECIFICATION structure element;

— a GLYPH INDEX MAP SPECIFICATION structure element;

— a COLOR SPACE SPECIFICATION structure element;

— a DATA SOURCE SPECIFICATION structure element;

— a PATTERN SPECIFICATION structure element;

— a FORM SPECIFICATION structure element.

There are no filter specification structure elements defined by this International Standard.

9.3.4 RESOURCE DEFINITION Processing

The processing of a RESOURCE DEFINITION structure element shall consist of:

a) verifying that the RESOURCE SPECIFICATION defines a resource of the resource class specified by the value of the immediately subordinate RESOURCE CLASS IDENTIFIER;

b) processing the immediately subordinate RESOURCE SPECIFICATION to define a resource;

c) adding the resource defined by the immediately subordinate RESOURCE SPECIFICATION to the environment of the Presentation Process for the scope of the RESOURCE DEFINITION;

d) binding the value of the ENVIRONMENT RESOURCE IDENTIFIER to the resource defined by the immediately subordinate RESOURCE SPECIFICATION for the scope of the RESOURCE DEFINITION.

A structure warning shall occur if the resource defined by the RESOURCE SPECIFICATION is not of the resource class identified by the value of the RESOURCE CLASS IDENTIFIER.

If the RESOURCE DEFINITION is subordinate to a DOCUMENT, a structure error shall occur if the resource defined by the RESOURCE SPECIFICATION cannot be added to the environment of the Presentation Process due to limitations in available space.

If there is a resource in the environment of the Presentation Process which is both

— of the resource class specified by the RESOURCE CLASS IDENTIFIER;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— identified in the environment of the Presentation Process by the value of the ENVIRONMENT RESOURCE IDENTIFIER;

then the binding of the value of the ENVIRONMENT RESOURCE IDENTIFIER to that resource shall be superseded by the binding defined in step d for the scope of the RESOURCE DEFINITION.

9.4 RESOURCE DECLARATION

A RESOURCE DECLARATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one RESOURCE CLASS IDENTIFIER structure element;

— one INTERNAL RESOURCE IDENTIFIER structure element;

— one ENVIRONMENT RESOURCE IDENTIFIER structure element.

The immediate subordinates shall occur in the above order.

9.4.1 INTERNAL RESOURCE IDENTIFIER

An INTERNAL RESOURCE IDENTIFIER structure element shall be a base structure element. The value of an INTERNAL RESOURCE IDENTIFIER shall be of type *Name*.

9.4.2 RESOURCE DECLARATION Processing

The processing of a RESOURCE DECLARATION structure element shall consist of:

a) identifying a declared resource as specified in 9.4.2.1;

b) binding the value of the immediately subordinate INTERNAL RESOURCE IDENTIFIER to a reference object as specified in 9.4.2.2;

c) adding the reference object and the resource to the Set of Declared Resources of the Block State of the most immediately superior BLOCK.

The processes of resource identification and name binding are specified below.

9.4.2.1 Resource Identification

If there is a RESOURCE DEFINITION in the context of the RESOURCE DECLARATION which both

— defines a resource of the resource class specified by the RESOURCE CLASS IDENTIFIER;

— binds an Environment Identifier which is the same as the value of the ENVIRONMENT RESOURCE IDENTIFIER to a reference object which references that resource;

then the declared resource shall be the resource defined by the most immediately superior such RESOURCE DEFINITION.

If there is no such RESOURCE DEFINITION, the resource which is to be declared shall be the resource (if any) in the environment of the Presentation Process which is both

— of the resource class specified by the RESOURCE CLASS IDENTIFIER;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642538 868

— identified in the environment of the Presentation Process by the value of the ENVIRONMENT RESOURCE IDENTIFIER.

9.4.2.2 Name Binding

The scope of the binding of the value of the INTERNAL RESOURCE IDENTIFIER to the reference object which references the declared resource is the set of all structure elements in the most immediately superior BLOCK which follow the RESOURCE DEFINITION in the sequential order and all of their subordinates, unless overridden by a subsequent RESOURCE DECLARATION in the same BLOCK or in a subordinate BLOCK.

If the RESOURCE DECLARATION falls within the scope of a preceding RESOURCE DECLARATION which bound the same Name to a reference object which references a resource of the same class, then the later binding shall take precedence within its own scope.

9.4.2.3 No Resource

If there is no resource which is of the class specified by the value of the RESOURCE CLASS IDENTIFIER and which is identified by the value of the ENVIRONMENT RESOURCE IDENTIFIER, no structure processing error or warning shall occur. However, no change shall be made to the Block State of the most immediately superior BLOCK, and the Name which is the value of the INTERNAL RESOURCE IDENTIFIER shall not be bound to a reference object by the processing of the RESOURCE DECLARATION. Consequently, a reference to a resource using that Name during the processing of the content of a TOKENSEQUENCE may result in a content processing error.

9.5 Context Dictionary

A Context Dictionary is a Dictionary which is defined by a DICTIONARY SPECIFICATION structure element. A Context Dictionary is not modifiable after processing of the DICTIONARY SPECIFICATION which defines it.

NOTE 36 Dictionary is defined in 18.2.1. The term "Context Dictionary" is not a generic term for any Dictionary on the Context Stack, but refers explicitly to a Dictionary created by DICTIONARY SPECIFICATION. When a distinction is necessary, the term "non-context Dictionary" shall be used to indicate any Dictionary that is not a Context Dictionary.

9.5.1 DICTIONARY SPECIFICATION

A DICTIONARY SPECIFICATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one or more TOKENSEQUENCE structure elements.

The immediate subordinates shall occur in the above order.

9.5.2 DICTIONARY SPECIFICATION Processing

The processing of a DICTIONARY SPECIFICATION structure element shall consist of:

a) Calling the Content Processor with:

— a content value which is obtained by concatenating the values of the TOKENSEQUENCE structure elements which are immediately subordinate to the DICTIONARY SPECIFICATION;

— a Context of Interpretation which is obtained from the Block State of the most immediately superior BLOCK as specified in 16.4.2;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642539 7T4

b) Upon completion of content processing, verifying that the element on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor is a DictionaryReference;

c) Making the Dictionary referenced by the DictionaryReference on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor read-only.

The Dictionary referenced by the DictionaryReference on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor shall be called the Context Dictionary defined by the RESOURCE SPECIFICATION.

A structure warning shall occur if element on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor is not a DictionaryReference, or if the status returned by the Content Processor is other than *OK*.

9.6 Color Space

A Color Space is represented by a Color Space Object, which is a Vector whose structure is as specified in clause 34.

9.6.1 Color Space Family Identifiers

Each Color Space is a member of a Color Space Family. Correspondingly, the first element of a Color Space Object is a Color Space Family Identifier. A Color Space Family Identifier is an OctetStringReference which references an OctetString which contains the canonical character string form of a public identifier (as defined by ISO/IEC 9070) which identifies the Color Space Family, encoded using ISO/IEC 646.

Each Presentation Process has a Color Space Family Identifier for each Color Space Family which it is capable of supporting. The Color Space Families defined by this International Standard, and the public identifier which is represented by the content of the OctetString referenced by each of the corresponding Color Space Family Identifiers, shall be shown in table 6.

Table 6

Color Space Family	Public Identifier (canonical character string form)
CIELAB	*ISO/IEC 10180//ColorSpaceFamily::CIELAB*
CIELUV	*ISO/IEC 10180//ColorSpaceFamily::CIELUV*
CIEBasedABC	*ISO/IEC 10180//ColorSpaceFamily::CIEBasedABC*
CIEBasedA	*ISO/IEC 10180//ColorSpaceFamily::CIEBasedA*
DeviceRGB	*ISO/IEC 10180//ColorSpaceFamily::DeviceRGB*
DeviceCMYK	*ISO/IEC 10180//ColorSpaceFamily::DeviceCMYK*
DeviceKX	*ISO/IEC 10180//ColorSpaceFamily::DeviceKX*
DeviceGray	*ISO/IEC10180//ColorSpaceFamily::DeviceGray*
Indexed	*ISO/IEC 10180//ColorSpaceFamily::Indexed*
NamedColor	*ISO/IEC 10180//ColorSpaceFamily::NamedColor*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9.6.2 COLOR SPACE SPECIFICATION

A COLOR SPACE SPECIFICATION structure element shall be a composite structure element whose immediate subordinates shall consist of:

— one COLOR SPACE FAMILY IDENTIFIER structure element;

— zero or one PRIMARY SET SPECIFICATION structure element;

— one or more TOKENSEQUENCE structure elements.

The immediate subordinates shall occur in the above order.

9.6.3 COLOR SPACE FAMILY IDENTIFIER

A COLOR SPACE FAMILY IDENTIFIER structure element shall be a base structure element. The value of a COLOR SPACE FAMILY IDENTIFIER shall be of type Public Object Identifier. The Object Names for which Public Object Identifier values are defined by this International Standard, and the Color Space Families identified by the corresponding Color Space Family Identifiers, are shown in table 7.

Table 7

Object Name	Color Space Family
ColorSpaceFamily::CIELAB	CIELAB
ColorSpaceFamily::CIELUV	CIELUV
ColorSpaceFamily::CIEBasedABC	CIEBasedABC
ColorSpaceFamily::CIEBasedA	CIEBasedA
ColorSpaceFamily::DeviceRGB	DeviceRGB
ColorSpaceFamily::DeviceCMYK	DeviceCMYK
ColorSpaceFamily::DeviceKX	DeviceKX
ColorSpaceFamily::DeviceGray	DeviceGray
ColorSpaceFamily::Indexed	Indexed
ColorSpaceFamily::NamedColor	NamedColor

9.6.4 PRIMARY SET SPECIFICATION

A PRIMARY SET SPECIFICATION provides a hint which characterizes color parameters referenced to the color space. The PRIMARY SET SPECIFICATION specifies the set of color primaries relative to which the color data has been generated. This information is useful for device color spaces, for which it may be used to affect the default mapping from parametric color specifications to device colorants. It may be ignored in processing the colorimetric (CIE based) color spaces.

A PRIMARY SET SPECIFICATION structure element shall be one of:

— a PRIMARY SET IDENTIFIER structure element;

— a PRIMARIES LIST structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9.6.5 PRIMARY SET IDENTIFIER

A PRIMARY SET IDENTIFIER structure element shall be a base structure element. The value of a PRIMARY SET IDENTIFIER shall be of type *Public Object Identifier*. There are no PRIMARY SET IDENTIFIER Public Object Identifier values defined by this International Standard.

9.6.6 PRIMARIES LIST

A PRIMARIES LIST structure element shall be a composite structure element which shall have the following immediate subordinates:

— one or more PRIMARY COLOR IDENTIFIER structure elements.

9.6.7 PRIMARY COLOR IDENTIFIER

A PRIMARY COLOR IDENTIFIER structure element shall be a base structure element. The value of a PRIMARY COLOR IDENTIFIER shall be of type *Public Object Identifier*. There are no PRIMARY COLOR IDENTIFIER Public Object Identifier values defined by this International Standard.

9.6.8 COLOR SPACE SPECIFICATION Processing

The processing of a COLOR SPACE SPECIFICATION structure element shall consist of:

a) Calling the Content Processor with:

   — a content value which is obtained by concatenating the values of the TOKENSEQUENCE structure elements which are immediately subordinate to the COLOR SPACE SPECIFICATION;

   — a Context of Interpretation which is obtained from the Block State of the most immediately superior BLOCK as specified in 16.4.2, with the exception of the replacement of the Operand Stack of the Virtual Machine State with an Operand Stack consisting of a single element which is the Color Space Family Identifier for the Color Space Family identified by the value of the COLOR SPACE FAMILY IDENTIFIER.

b) Upon completion of content processing, verifying that the element on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor is a VectorReference;

c) Verifying that the element of the referenced Vector corresponding to the index value of zero is the Color Space Family Identifier which was on the top of the Operand Stack of the Virtual Machine state of the call to the Content Processor.

The Vector referenced by the VectorReference which was on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor (or the VectorReference itself) shall be called the Color Space Object defined by the COLOR SPACE SPECIFICATION.

A structure warning shall occur if element on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor is not a VectorReference, if the referenced Vector is not a Color Space Object as defined by clause 34, if the element of the referenced Vector corresponding to the index value of zero is not the same as the OctetStringReference on the top of the Operand Stack of the Virtual Machine state of the call to the Content Processor, or if the status returned by the Content Processor is other than *OK*.

9.6.9 Standard Color Space Resources

This International Standard defines *Public Object Identifier* values which identify several Color Space resources The Vector referenced by the VectorReference for each of these Color Spaces consists of a single

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

element which is the Color Space Family Identifier for the Color Space Family. The Object Names for which *Public Object Identifier* values are defined by this International Standard, and the Color Space Families of the Color Spaces identified by those values, are shown in table 8.

Table 8

Object Name	Color Space Family
ColorSpace::DeviceRGB	DeviceRGB
ColorSpace::DeviceCMYK	DeviceCMYK
ColorSpace::DeviceKX	DeviceKX
ColorSpace::DeviceGray	DeviceGray

9.7 Data Source

A Data Source is represented by a *StreamObject* as defined in clause 31.

9.7.1 DATA SOURCE SPECIFICATION

A DATA SOURCE SPECIFICATION shall be one of

— a LOCATION IDENTIFIER structure element;

— a DATA BLOCK structure element.

9.7.2 DATA BLOCK

A DATA BLOCK structure element shall be a base structure element The value of a DATA BLOCK shall be of type *Octet String*

The value of the DATA BLOCK structure element represents a sequence of octets. The sequence of octets represented by the depends on the Structure Representation and Interchange Format being used:

— the value of the DATA BLOCK structure element represents the sequence of octets encoded using the ASCII85 encoding if the Clear Text Structure Representation and Interchange Format is being used;

— the value of the DATA BLOCK structure element represents the sequence of octets themselves if the Binary Structure Representation and Interchange Format is being used.

The determination of the sequence of octets represented by the value of the DATA BLOCK structure element is performed implicitly as part of the creation of the StreamObject defined by the DATA SOURCE SPECIFICATION structure element.

9.7.3 DATA SOURCE SPECIFICATION Processing

9.7.3.1 LOCATION IDENTIFIER DATA SOURCE SPECIFICATION Processing

The processing of a DATA SOURCE SPECIFICATION which is a LOCATION IDENTIFIER shall consist of:

a) identifying the data block which is identified by the LOCATION IDENTIFIER;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

b) defining a Data Source resource for which the data consists of the sequence of octets which comprise the data block identified in step a, followed by an end-of-data indication.

The Data Source resource defined in step b shall be called the Data Source resource defined by the DATA SOURCE SPECIFICATION.

9.7.3.2 DATA BLOCK DATA SOURCE SPECIFICATION Processing

The processing of a DATA SOURCE SPECIFICATION which is a DATA BLOCK shall consist of defining a Data Source resource for which the data consists of a sequence of octets which are derived from the value of the immediately subordinate DATA BLOCK structure element, followed by an end-of-data indication. The sequence of octets depends on the Structure Representation and Interchange Format being used.

— If the DATA BLOCK structure element is encoded using the Clear Text Structure Representation and Interchange Format, the sequence of octets consists of the result of applying the ASCII85 filter defined in 31.2.2 to the sequence of octets which comprise the value of the DATA BLOCK structure element.

— If the DATA BLOCK structure element is encoded using the Binary Structure Representation and Interchange Format, the sequence of octets consists of the value of the DATA BLOCK structure element itself.

9.7.4 Standard Data Source Resources

This International Standard defines one *Public Object Identifier* value which identifies a Data Source resource. The Object Name for which the *Public Object Identifier* value is defined by this International Standard, and the Data Source identified by that value is shown in table 9.

Table 9

Object Name	Data Source
DataSource::Document	Document

The Data Source resource *DataSource::Document* represents data which is contained in a token in the content value which is being processed by the Content Processor as defined in clause 28.

9.8 Filter

A Filter resource is represented by a Filter Identifier. A Filter Identifier is an *OctetStringReference* which references an OctetString which contains the canonical character string form of a public identifier (as defined by ISO/IEC 9070) which identifies the Filter. The Filter resources defined by this International Standard, and the public identifier which is represented by the content of the OctetString referenced by each of the corresponding Filter Identifiers, is shown in table 10.

Table 10

Filter	Public Identifier (canonical character string form)
ASCIIHexDecode	*ISO/IEC 10180//Filter::ASCIIHexDecode*
ASCII85Decode	*ISO/IEC 10180//Filter::ASCII85Decode*
RunLengthDecode	*ISO/IEC 10180//Filter::RunLengthDecode*
CCITTFaxDecode	*ISO/IEC 10180//Filter::CCITTFaxDecode*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Filter	Public Identifier (canonical character string form)
NullDecode	*ISO/IEC 10180//Filter::NullDecode*

9.8.1 Filter RESOURCE SPECIFICATION

There is no RESOURCE SPECIFICATION structure element defined by this International Standard which defines a Filter resource. The process by which a Filter resource is installed in the environment of the Presentation Process is outside of the scope of this International Standard.

9.8.2 Standard Filters

This International Standard defines a *Public Object Identifier* value which identifies a Filter resource for each of the Filters defined in clause 31. The Object Names for which *Public Object Identifier* values are defined by this International Standard, and the Filter resources identified by those values, are ashown in table 11.

Table 11

Object Name	Filter
Filter::ASCIIHexDecode	ASCIIHexDecode
Filter::ASCII85Decode	ASCII85Decode
Filter::RunLengthDecode	RunLengthDecode
Filter::CCITTFaxDecode	CCITTFaxDecode
Filter::NullDecode	NullDecode

9.9 Pattern

A Pattern is represented by a Pattern Dictionary as defined in clause 32.

9.9.1 PATTERN SPECIFICATION

The value of a PATTERN SPECIFICATION shall be a composite structure element whose immediate subordinates shall consist of:

— one or more TOKENSEQUENCE structure elements.

9.9.2 PATTERN SPECIFICATION Processing

The processing of a PATTERN SPECIFICATION shall consist of:

a) Calling the Content Processor with:

— a content value which is obtained by concatenating the values of the TOKENSEQUENCE structure elements which are immediately subordinate to the PATTERN SPECIFICATION;

— a Context of Interpretation which is obtained from the Block State of the most immediately superior BLOCK as specified in 16 4.2;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642545 TT8

b) Upon completion of content processing, verifying that the element on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor is a DictionaryReference.

The Dictionary which was referenced by the DictionaryReference on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor shall be called the Pattern Dictionary defined by the PATTERN SPECIFICATION.

A structure warning shall occur if the element on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor is not a DictionaryReference, or if the status returned by the Content Processor is other than *OK*. The Structure Processor may but need not raise a structure warning if the Pattern Object defined by the PATTERN SPECIFICATION is not a valid Pattern Dictionary.

9.10 Form

A Form is represented by a Form Dictionary as defined in clause 33.

9.10.1 FORM SPECIFICATION

A FORM SPECIFICATION shall be a composite structure element whose immediate subordinates shall consist of:

— one or more TOKENSEQUENCE structure elements.

9.10.2 FORM SPECIFICATION Processing

The processing of a FORM SPECIFICATION shall consist of:

a) Calling the Content Processor with:

— a content value which is obtained by concatenating the values of the TOKENSEQUENCE structure elements which are immediately subordinate to the FORM SPECIFICATION;

— a Context of Interpretation which is obtained from the Block State of the most immediately superior BLOCK as specified in 16.4.2;

b) Upon completion of content processing, verifying that the element on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor is a DictionaryReference.

The Dictionary which was referenced by the DictionaryReference on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor shall be called the Form Dictionary defined by the FORM SPECIFICATION.

A structure warning shall occur if the element on the top of the Operand Stack of the Virtual Machine State returned by the Content Processor is not a DictionaryReference, or if the status returned by the Content Processor is other than *OK*. The Structure Processor may but need not raise a structure warning if the Form Object defined by the FORM SPECIFICATION is not a valid Form Dictionary.

10 Fonts

Fonts are the information objects which are used to convey the graphical elements used for the presentation of character text.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642546 934

10.1 General

The SPDL font architecture is based on ISO/IEC 9541. The following are the major elements of that architecture.

10.1.1 Font Resources

The information object defined by ISO/IEC 9541 for the interchange of font information is the *Font Resource*. A Font Resource includes glyph shape information and metric information. The graphical elements which are specified by a Font Resource are called glyph representations. Font Resource architecture and interchange formats are defined in ISO/IEC 9541.

A Font Resource which conforms to ISO/IEC 9541-1 may be interchanged as either an ASN.1 data structure or an SGML data structure as defined by ISO/IEC 9541-2 and ISO/IEC 9541-3. A Font Resource may also be placed in the environment of the Presentation Process by other means.

The information in a Font Resource which is used for presenting glyph representations is represented in SPDL as part of an Indexed Font Dictionary.

10.1.2 Glyph Index Maps

Glyph representations in a Font Resource which conforms to ISO/IEC 9541 are identified by "structured-names" as defined by ISO/IEC 9541. For reasons of efficiency, this International Standard defines a data structure which provides for the access of glyph representations and relevant metric information with a mapping which enables access by means of Cardinal numbers called glyph index values. The mapping of glyph index values to glyph identifiers is accomplished by a *Glyph Index Map*.

NOTE 37 Data Structures in which glyph representations are identified by numeric values correspond to Indexed Fonts rather than Font Resources.

10.1.3 Indexed Fonts

An *Indexed Font* provides the information needed for presenting glyph representations from a Font Resource by the Presentation Process. Each Indexed Font includes glyph shape information and metrics and a Glyph Index Map. This information need not be separable

An Indexed Font may occur as a resource in the environment of the Presentation Process, or it may be specified by a FONT SPECIFICATION.

An Indexed Font is represented and specified in SPDL by an Indexed Font Dictionary as defined in clause 24. Data structures containing glyph shape representations which are accessed by Integer values rather than by Glyph Identifiers correspond to Indexed Fonts rather than to Font Resources.

10.1.4 Other Indexed Fonts

In addition to those Indexed Fonts which are constructed by FONT SPECIFICATION structure elements, a Presentation Process may use Indexed Fonts which have been installed in the environment of the Presentation Process by methods which are outside of the scope of this International Standard. The use of such Indexed Fonts is specifically enabled by this International Standard.

Indexed Fonts which are not defined according to the ISO/IEC 9541 architecture may not have the internal structure defined by clause 24 (for example, there may be no explicit glyph identifiers). Such Indexed Fonts would be represented by "execute only" Indexed Font Dictionaries

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642547 870

10.2 Representation of Glyph Identifiers

Glyph Identifiers occurring in SPDL structure are represented by values of the *Glyph Identifier* type. *Glyph Identifier* values may also occur in

— Font Resources interchanged using the font information interchange format defined by ISO/IEC 9541-2;

— Font Resources, Indexed Fonts, or Glyph Index Maps which are installed as resources in the environment of an SPDL Presentation Process.

In the processing of SPDL structure, and in the installation of Font Resources, Indexed Fonts, or Glyph Index Maps in the environment of the Presentation Process, *Glyph Identifier* values are converted into *Identifier* values which are used in the representation of these resources as Dictionaries. Two *Glyph Identifier* values are converted to the same *Identifier* value if and only if they are equal as defined in 6 16.5. The conversion of *Glyph Identifier* values to the *Identifier* values which occur in these Dictionaries is specified in this clause.

10.2.1 Conversion of ISO10036 Glyph Names

Each ISO10036 Glyph Name is semantically equivalent to a public identifier for which the canonical character string form is of the form "ISO/IEC 10036/RA//Glyphs::*nnnn*". The ISO10036 Glyph Name which is equivalent to "ISO/IEC 10036/RA//Glyphs::*nnnn*" is converted to the Identifier which is the result of

{*glyphnameref* **ConvertToIdentifier**}

where *glyphnameref* is an OctetStringReference which references an OctetString which represents the character string "afii*nnnn*", encoded using ISO 646 IRV.

NOTE 38 The resulting Identifier is equivalent to the Name "afii*nnnn*".

10.2.2 Conversion of Other Structured Glyph Names

A Structured Glyph Name which is not an ISO10036 Glyph Name is converted to the Identifier which is the result of

{*structurednameref* **ConvertToIdentifier**}

where *structurednameref* is a is an OctetStringReference which references an OctetString which represents the canonical character string form of the public identifier which is semantically equivalent to the Structured Glyph Name, encoded using ISO 646 IRV.

NOTE 39 The resulting Identifier is not equivalent to any Name.

10.2.3 Conversion of Simple Glyph Names

A Simple Glyph Name is converted to the Identifier which is the result of

{*glyphnameref* **ConvertToIdentifier**}

where *glyphnameref* is a is an OctetStringReference which references an OctetString which represents the Simple Glyph Name, encoded using ISO 646 IRV.

NOTES

40 The resulting Identifier is equivalent to the Name which is specified by the same character string as the Simple Glyph Name.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642548 707

41 The AFII Glyph Name "afii*nnnn*" and the ISO10036 Glyph Name "ISO/IEC 10036/RA//Glyphs::*nnnn*" are equal as a Glyph Identifiers, and are also converted to the same Identifier, the Name "afii*nnnn*".

10.3 Font References

A Font Resource in the environment of the Presentation Process may be referenced for use in document presentation by a Font Reference. A Font Reference may identify a Font Resource by either or both of:

– a Font Resource Identifier;

— a set of properties which characterize the Font Resource.

In addition, a Font Reference can specify a *match rules* value, which specifies how the Presentation Process determines whether Font Resource property matches a property specified by a Font Reference, and a *satisfaction criterion* value, which specifies how the Presentation Process determines whether a Font Resource satisfies a Font Reference.

NOTE 42 The environment of the Presentation Process may be local or may include other processes (e.g. a Font Service) which may be accessed by the Presentation Process.

10.3.1 Font Resource Identifiers

A Font Resource in the environment of the Presentation Process is identified by one or more Font Resource Identifiers. A Font Resource Identifier may be a Public Identifier, an Object Identifier, or an Environment Name.

If the Font Resource is an ISO/IEC 9541 Font Resource, the value of the font resource name property shall be one of the Font Resource Identifiers which identifies the Font Resource in the environment of the Presentation Process.

NOTE 43 The font resource name property is named by the structured-name which is semantically equivalent to the public identifier whose canonical character string form is ISO/IEC 9541-1//FONTNAME.

In addition to the font name property (if any), one or more Environment Identifiers may be assigned to a Font Resource as Font Resource Identifiers when it is installed in the environment of the Presentation Process. The assignment of Font Resource Identifiers other than the font resource name property to a Font Resource is outside of the scope of this International Standard.

10.3.2 Font Properties

The font properties which can be specified by a Font Reference consist of a set of *required properties* and a set of *advisory properties* The font properties are specified by Font Attribute Sets as defined by ISO/IEC 9541.

NOTE 44 The advisory properties may be used to select a Font Resource from several Font Resources in the environment of the Presentation Process which satisfy the required properties; or to select a substitute Font Resource when there is no Font Resources in the environment of the Presentation Process which satisfies the required properties. The Presentation Process may also ignore the advisory properties in some or all instances of Font Resource selection.

NOTE 45 The process of selecting a substitute Font Resource when no Font Resource in the environment of the Presentation Process satisfies the required properties may but need not involve the use of advisory properties. The system may choose to use a single system dependent substitute Font Resource in all such cases.

10.3.3 Unsatisfied Font References

Processing a Font Reference always results if the selection of a Font Resource for use in document presentation No error or warning occurs if the selected Font Resource fails to satisfy the Font Reference: however, a

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642549 643

content warning occurs if content processing executes **FindFont** with an argument which is bound to an Indexed Font for which the Font Resource fails to satisfy the Font Reference.

10.4 FONT SPECIFICATION

An Indexed Font is specified by a FONT SPECIFICATION structure element. A FONT SPECIFICATION structure element shall be one of the following:

— a FONT IDENTIFIER FONT SPECIFICATION structure element;

— a FONT REFERENCE FONT SPECIFICATION structure element;

— a FONTTYPE1 FONT SPECIFICATION structure element;

— a REMAPPED FONT SPECIFICATION structure element;

— a COMPOSITE FONT SPECIFICATION structure element;

— a USER FONT SPECIFICATION structure element.

NOTE 46 The FONT SPECIFICATION definition is recursive.

10.5 FONT IDENTIFIER FONT SPECIFICATION

A FONT IDENTIFIER FONT SPECIFICATION structure element shall be an ENVIRONMENT RESOURCE IDENTIFIER structure element as defined in 9.3.1.

The processing of a FONT IDENTIFIER FONT SPECIFICATION shall consist of identifying the Indexed Font in the environment of the Presentation Process which is identified by the value of the FONT IDENTIFIER FONT SPECIFICATION structure element. If there is no Indexed Font in the environment of the Presentation Process which is identified by the value of the FONT IDENTIFIER FONT SPECIFICATION, a system dependent Indexed Font shall be selected.

The Indexed Font resource defined by the FONT IDENTIFIER FONT SPECIFICATION shall be the Indexed Font identified or selected as specified in the above paragraph.

10.6 FONT REFERENCE FONT SPECIFICATION

A FONT REFERENCE FONT SPECIFICATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one FONT REFERENCE structure element;

— one GLYPH INDEX MAP IDENTIFIER structure element.

The immediate subordinates shall occur in the above order.

10.6.1 GLYPH INDEX MAP IDENTIFIER

A GLYPH INDEX MAP IDENTIFIER structure element shall be one of:

— ENVIRONMENT RESOURCE IDENTIFIER

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642550 365

10.6.2 FONT REFERENCE FONT SPECIFICATION Processing

The processing of a FONT REFERENCE FONT SPECIFICATION shall consist of the following:

a) Selecting a Font Resource in the environment of the Presentation Process by processing the FONT REFERENCE as specified in 10.7.

b) Selecting the Glyph Index Map in the environment of the Presentation Process which is identified by the value of the GLYPH INDEX MAP IDENTIFIER.

   — If the there is no Glyph Index Map in the environment of the Presentation Process which is identified by the value of the GLYPH INDEX MAP IDENTIFIER, a system dependent Glyph Index Map shall be used.

c) Constructing an Indexed Font Dictionary by combining the Font Resource identified in the first step with the Glyph Index Map identified in the second step.

The Indexed Font resource defined by FONT REFERENCE FONT SPECIFICATION shall be the Indexed Font defined by the Indexed Font Dictionary constructed in step c.

NOTE 47 If the referenced Font Resource or identified Glyph Index Map cannot be found, the substituted Font Resource may contain no glyphs at all which are identified by the Glyph Identifiers in the Glyph Index Map used, or the glyph images generated may not be those intended by the document creator.

10.7 FONT REFERENCE

A FONT REFERENCE identifies a Font Resource by means of a Font Resource Identifier and, optionally, a supplementary set of font properties. FONT REFERENCE processing consists of selecting a Font Resource on the basis of the FONT REFERENCE structure element and identifying whether the selected Font resource satisfies the FONT REFERENCE.

NOTE 48 The environment of the Presentation Process may be local or may include other processes (e.g. a Font Service) which may be accessed by the Presentation Process.

A FONT REFERENCE structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one FONT RESOURCE IDENTIFIER structure element;

— zero or one REQUIRED PROPERTIES structure element;

— zero or one ADVISORY PROPERTIES structure element;

— zero or one MATCH RULES structure element;

— zero or one SATISFACTION CRITERION structure element.

The immediate subordinates shall occur in the above order.

10.7.1 FONT RESOURCE IDENTIFIER

A FONT RESOURCE IDENTIFIER structure element shall be an ENVIRONMENT RESOURCE IDENTIFIER structure element as defined in 9.3.1.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642551 2T1

10.7.2 REQUIRED PROPERTIES

A REQUIRED PROPERTIES structure element shall be a Font Attribute Set as defined by ISO/IEC 9541. The *required properties* of the FONT REFERENCE shall be the set of properties specified by the REQUIRED PROPERTIES structure element.

10.7.3 ADVISORY PROPERTIES

An ADVISORY PROPERTIES structure element shall be a Font Attribute Set as defined by ISO/IEC 9541. The *advisory properties* of the FONT REFERENCE shall be the set of properties specified by the ADVISORY PROPERTIES structure element.

10.7.4 MATCH RULES

A MATCH RULES structure element shall be a base structure element The value of a MATCH RULES structure element shall be of type *Public Object Identifier*.

The value of a MATCH RULES structure element specifies the criterion which shall be used to determine whether a Font Resource satisfies each of the property specifications in the required properties or the advisory properties. The Object Names for which Public Object Identifier values are defined by this International Standard, and the meaning of those values, are shown in table 12.

Table 12

Object Name	Meaning
FontReference::MatchRules::Same	A Font Resource satisfies a property if a value is specified for the property in the Font Resource, and if that value is the same as the value of the corresponding required or advisory property.
FontReference::MatchRules::SameIfSpecified	A Font Resource satisfies a property if either no value is specified for the property for the Font Resource, or if the value of the property specified in the Font Resource is the same as the value of the corresponding required or advisory property.

The default value of the MATCH RULES structure element is *FontReference::MatchRules::Same*.

10.7.5 SATISFACTION CRITERION

A SATISFACTION CRITERION structure element shall be a base structure element. The value of a SATISFACTION CRITERION structure element shall be of type *Public Object Identifier*.

The Object Names for which *Public Object Identifier* values are defined by this International Standard, and the meaning of those values, are shown in table 13.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642552 138

Table 13

Object Name	Meaning
FontReference::Satisfaction::Name	A Font Resource satisfies the FONT REFERENCE only if the Font Resource is identified by the value of the FONT RESOURCE IDENTIFIER. If there is no FONT RESOURCE IDENTIFIER, no Font Resource satisfies the FONT REFERENCE
FontReference::Satisfaction::NameOrProps	A Font Resource satisfies the FONT REFERENCE if the Font Resource is identified by the value of the FONT RESOURCE IDENTIFIER or if it satisfies all of the required properties.
FontReference::Satisfaction::Any	Any Font Resource satisfies the FONT REFERENCE.

The default value of the SATISFACTION CRITERION structure element is *FontReference::Satisfaction::NameOrProps*.

10.7.6 FONT REFERENCE Processing

FONT REFERENCE processing consists of

a) selecting a Font Resource;

b) deciding whether the selected Font Resource satisfies the FONT REFERENCE.

The procedure used for selecting a Font Resource shall be as follows:

a) If there is a Font Resource in the environment of the Presentation Process which is identified by the Font Resource Identifier, that Font Resource shall be selected.

b) If there is no Font Resource in the environment of the Presentation Process which is identified by the Font Resource Identifier, identify the list of Font Resources in the environment of the Presentation Process which satisfies the required properties.

   A Font Resource satisfies the required properties if it satisfies each of the properties as specified by the value of the MATCH RULES structure element.

c) If there is exactly one Font Resource in the environment of the Presentation Process which satisfies the required properties, that Font Resource shall be selected.

d) If there is more than one Font Resource in the environment of the Presentation Process which satisfies the required properties, a selection shall be made from among them. This selection may depend on the advisory properties

e) If there is no Font Resource in the environment of the Presentation Process which satisfies the required properties, a substitute Font Resource shall be selected in a system dependent manner. The selection of the substitute Font Resource may depend on any combination of the required properties and the advisory properties.

Whether the Font Resource which is selected satisfies the FONT REFERENCE depends on the value of the SATISFACTION CRITERION structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642553 074

10.7.7 Standard FONT RESOURCE IDENTIFIER Values

This International Standard defines twelve *Public Object Identifier* values for the identification of Font Resources. For each of these values, there is a Font Resource in the environment of the Presentation Process which satisfies a FONT REFERENCE consisting solely of a FONT RESOURCE IDENTIFIER whose value is the *Public Object Identifier* value. The Object Names for which *Public Object Identifier* values are defined by this International Standard, and the characteristics of the Font Resources identified by those values, are shown in table 14.

Table 14

Object Name	Font Resource Characteristics
Fonts::ISO-Serif::Regular	a regular weight serif Font Resource
Fonts::ISO-Serif::Bold	a bold weight serif Font Resource
Fonts::ISO-Serif::Italic	a regular weight italic serif Font Resource
Fonts::ISO-Serif::BoldItalic	a bold weight italic serif Font Resource
Fonts::ISO-SanSerif::Regular	a regular weight sanserif Font Resource
Fonts::ISO-SanSerif::Bold	a bold weight sanserif Font Resource
Fonts::ISO-SanSerif::Italic	a regular weight italic sanserif Font Resource
Fonts::ISO-SanSerif::BoldItalic	a bold weight italic sanserif Font Resource
Fonts::ISO-Monospace::Regular	a regular weight monospaced Font Resource
Fonts::ISO-Monospace::Bold	a bold weight monospaced Font Resource
Fonts::ISO-Monospace::Italic	a regular weight italic monospaced Font Resource
Fonts::ISO-Monospace::BoldItalic	a bold weight italic monospaced Font Resource

The specific properties of the Font Resources identified by these values are specified in annex D.

NOTE 49 These properties include the glyph complement, the escapements for each glyph, and properties which specify the typeface classification. They do not include the specific specific shapes represented by the glyph shape representations.

10.8 FONTTYPE1 FONT SPECIFICATION

A FONTTYPE1 FONT SPECIFICATION structure element shall be a base structure element. The value of a FONTTYPE1 FONT SPECIFICATION shall be of type *Octet String*. The value of a FONTTYPE1 FONT SPECIFICATION shall be an instance of the Indexed Font Representation Format specified in annex C of this International Standard.

The processing of a FONTTYPE1 FONT SPECIFICATION shall consist of interpreting the value of the FONTTYPE1 FONT SPECIFICATION to define an Indexed Font as specified in annex C of this International Standard. The resulting Indexed Font shall be the Indexed Font resource defined by the FONTTYPE1 FONT SPECIFICATION.

10.9 REMAPPED FONT SPECIFICATION

A REMAPPED FONT SPECIFICATION structure element shall be a composite structure element which shall have the following immediate subordinates:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— one FONT IDENTIFIER FONT SPECIFICATION structure element;

— zero or one REMAP structure element.

The immediate subordinates shall occur in the above order.

10.9.1 REMAP

A REMAP structure element shall be one of:

— a GLYPH INDEX MAP IDENTIFIER structure element;

— a GLYPH INDEX MAP TRANSLATION TABLE structure element.

10.9.2 GLYPH INDEX TRANSLATION TABLE

A GLYPH INDEX TRANSLATION TABLE structure element shall be a composite structure element which shall have the following immediate subordinates:

— one TABLE SIZE structure element;

— one or more GLYPH INDEX structure elements.

The immediate subordinates shall occur in the above order.

10.9.2.1 TABLE SIZE

A TABLE SIZE structure element shall be a base structure element. The value of a TABLE SIZE structure element shall be of type *Cardinal*.

10.9.2.2 GLYPH INDEX

A GLYPH INDEX structure element shall be a base structure element. The value of a GLYPH INDEX structure element shall be of type *Cardinal*.

The number of GLYPH INDEX structure elements shall be the same as the value of TABLE SIZE. The value of each GLYPH INDEX shall be a Cardinal between 0 and the value of TABLE SIZE minus 1.

10.9.2.3 GLYPH INDEX TRANSLATION TABLE Processing

The effect of applying a GLYPH INDEX TRANSLATION TABLE to a Glyph Index Map shall consist of creating a new Encoding Vector as follows:

> For each index value n between 0 and the value of TABLE SIZE minus 1, the value of the element of the new Encoding Vector whose index is n shall be the value of the element of the old Encoding Vector whose index is the value of the nth GLYPH INDEX in the GLYPH INDEX TRANSLATION TABLE.

10.9.3 REMAPPED FONT SPECIFICATION Processing

The processing of the REMAPPED FONT SPECIFICATION shall consist of the following.

a) Processing the FONT IDENTIFIER FONT SPECIFICATION as specified in 10.5 to obtain an Indexed Font.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642555 947

b) If the REMAP structure element is a GLYPH INDEX MAP IDENTIFIER, selecting the Glyph Index Map in the environment of the Presentation Process which is identified by the value of the GLYPH INDEX MAP IDENTIFIER.

   — If the there is no Glyph Index Map in the environment of the Presentation Process which is identified by the value of the GLYPH INDEX MAP IDENTIFIER, a system dependent Glyph Index Map shall be used.

c) If the REMAP structure element is a GLYPH INDEX MAP TRANSLATION TABLE, creating a new Glyph Index Map from the Glyph Index Map of the Indexed Font obtained in step a as specified in 10.9.2.

d) Constructing an Indexed Font Dictionary by replacing the Glyph Index Map in a copy of the Indexed Font Dictionary which represents the Indexed Font identified in step a with the Glyph Index Map selected in step b or created in step c.

The Indexed Font resource defined by REMAPPED FONT SPECIFICATION shall be the Indexed Font defined by the Indexed Font Dictionary constructed in step d.

10.9.4 Errors and Warnings

If there is no Indexed Font in the environment of the Presentation Process which is identified by the value of the FONT IDENTIFIER FONT SPECIFICATION, if the identified Indexed Font is a composite Indexed Font rather than a base Indexed Font, or if there is no Glyph Index Map in the environment of the Presentation Process which is identified by the value of the GLYPH INDEX MAP IDENTIFIER, a structure warning shall occur.

NOTE 50 If the intended Indexed Font or Glyph Index Map cannot be found, the selected Indexed Font may contain no glyphs at all which are identified by the Glyph Identifiers in the Glyph Index Map used, or the glyph images generated may not be those intended by the document creator.

10.10 COMPOSITE FONT SPECIFICATION

A COMPOSITE FONT SPECIFICATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one FMAPTYPE structure element;

— zero or one FMAPTYPE PARAMETERS LIST structure element;

— one FONT INDEX MAP structure element;

— one FONT LIST structure element.

The immediate subordinates shall occur in the above order. If the value of FMAPTYPE is 6, the FMAPTYPE PARAMETERS LIST structure element shall be present

10.10.1 FMAPTYPE

An FMAPTYPE structure element shall be a base structure element. The value of an FMAPTYPE structure element shall be of type *Cardinal*. The value shall be one of the values which identify one of the Composite Font Glyph Mapping algorithms defined in clause 26.

10.10.2 FMAPTYPE PARAMETERS LIST

An FMAPTYPE PARAMETERS LIST structure element shall be a composite structure element which shall have the following immediate subordinates:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642556 883

— zero or one ESCCHAR structure element;

— zero or one SHIFTOUT structure element;

— zero or one SHIFTIN structure element;

— zero or one SUBSVECTOR structure element.

The immediate subordinates may occur in any order. If the value of FMAPTYPE is 6, the SUBSVECTOR structure element shall be present.

10.10.2.1 ESCCHAR

An ESCCHAR structure element shall be a base structure element. The value of an ESCCHAR structure element shall be of type *Cardinal*. The value of an ESCCHAR structure element shall be between 0 and 255 inclusive.

10.10.2.2 SHIFTOUT

A SHIFTOUT structure element shall be a base structure element. The value of a SHIFTOUT structure element shall be of type *Cardinal*. The value of a SHIFTOUT structure element shall be between 0 and 255 inclusive.

10.10.2.3 SHIFTIN

A SHIFTIN structure element shall be a base structure element. The value of a SHIFTIN structure element shall be of type *Cardinal*. The value of a SHIFTIN structure element shall be between 0 and 255 inclusive.

10.10.2.4 SUBSVECTOR

A SUBSVECTOR structure element shall be a base structure element. The value of a SUBSVECTOR structure element shall be of type *Octet String*.

10.10.3 FONT INDEX MAP

A FONT INDEX MAP structure element shall be a composite structure element whose immediate subordinates shall consist of:

— one FONT INDEX MAP SIZE structure element;

— one or more INDEX VALUE structure elements.

The immediate subordinates shall occur in the above order

10.10.3.1 FONT INDEX MAP SIZE

A FONT INDEX MAP SIZE structure element shall be a base structure element. The value of a FONT INDEX MAP SIZE structure element shall be of type *Cardinal*

10.10.3.2 INDEX VALUE

An INDEX VALUE structure element shall be a base structure element. The value of an INDEX VALUE structure element shall be of type *Cardinal*. The number of INDEX VALUE structure elements shall be the same as the value of the MAP SIZE structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

10.10.3.3 FONT INDEX MAP Processing

The processing of a FONT INDEX MAP shall consist of creating an Encoding Vector, as defined in clause 26, whose elements have the same values as the values of the corresponding INDEX VALUE structure elements. The value of the first INDEX VALUE becomes the first element of the Vector, the value of the second INDEX VALUE becomes the second element of the Vector, etc..

10.10.4 FONT LIST

A FONT LIST structure element shall be a composite structure element which shall have the following immediate subordinates:

— one or more FONT SPECIFICATION structure elements.

10.10.5 COMPOSITE FONT SPECIFICATION Processing

The processing of a COMPOSITE FONT SPECIFICATION shall consist of creating a Composite Font Dictionary, as defined in clause 26, for which

— the value of FontType is <0: *Integer*>;

— the value of FontMatrix is the identity transformation;

— the value of FMapType is the value of the FMAPTYPE structure element;

— the value of EscChar is the value of the value of the ESCCHAR structure element which is subordinate to the subordinate FMAPTYPE PARAMETERS LIST if present, or equal to 255 if no such structure element is present;

— the value of ShiftOut is the value of the value of the SHIFTOUT structure element which is subordinate to the subordinate FMAPTYPE PARAMETERS LIST if present, or equal to 14 if no such structure element is present;

— the value of ShiftIn is the value of the value of the SHIFTIN structure element which is subordinate to the subordinate FMAPTYPE PARAMETERS LIST if present, or equal to 15 if no such structure element is present;

— the value of SubsVector is the value of the value of the SUBSVECTOR structure element which is subordinate to the subordinate FMAPTYPE PARAMETERS LIST if present, or is not present if no such structure element is present;

— the value of an Encoding is a VectorReference which references the Encoding Vector defined by the immediately subordinate FONT INDEX MAP as specified in 10.10.3;

— the value of FDepVector is a reference to a Vector of Indexed Fonts, where the Indexed Fonts are generated by processing each of the FONT SPECIFICATION structure elements which are immediately subordinate to the FONT LIST, in turn.

The Indexed Font defined by the COMPOSITE FONT SPECIFICATION shall be the Indexed Font defined by this Composite Font Dictionary.

NOTE 51 If the number of FONT SPECIFICATION structure elements subordinate to the FONT LIST is less than the maximum index value in the identified Font Index Map, then Content Processing errors may occur if larger Font Index values are generated during the use of the Composite Indexed Font, as specified in clause 26.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642558 656

10.11 USER FONT SPECIFICATION

A USER FONT SPECIFICATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one or more TOKENSEQUENCE structure elements.

The processing of a USER FONT SPECIFICATION shall consist of:

a) Calling the Content Processor with:

   — a content value which is obtained by concatenating the values of the TOKENSEQUENCE structure elements which are immediately subordinate to the USER FONT SPECIFICATION;

   — a Context of Interpretation which is obtained from the Block State of the most immediately superior BLOCK as specified in 16.4.2;

b) Upon completion of content processing, verifying that the element on the top of the Operand Stack of the Virtual Machine State of the Context of Interpretation returned by the Content Processor is a DictionaryReference.

The Indexed Font defined by USER FONT SPECIFICATION shall be the Indexed Font defined by the Indexed Font Dictionary defined in step b.

A structure error shall occur if element on the top of the Operand Stack of the virtual machine state of the Context of Interpretation returned by the Content Processor is not a *DictionaryReference*, or if the status returned by the Content Processor is other than *OK*.

10.12 GLYPH INDEX MAP SPECIFICATION

A GLYPH INDEX MAP SPECIFICATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one GLYPH INDEX MAP SIZE structure element;

— one GLYPH IDENTIFIER LIST structure element.

The immediate subordinates shall occur in the above order.

10.12.1 GLYPH INDEX MAP SIZE

A GLYPH INDEX MAP SIZE structure element shall be a base structure element. The value of a GLYPH INDEX MAP SIZE structure element shall be of type *Cardinal*.

10.12.2 GLYPH IDENTIFIER LIST

The GLYPH IDENTIFIER LIST structure element shall be a composite structure element which shall have the following immediate subordinates:

— one or more GLYPH IDENTIFIER structure elements.

The size of the GLYPH IDENTIFIER LIST shall be the same as the value of the most immediately superior MAP SIZE structure element. The value of each GLYPH IDENTIFIER structure element shall be of type *Glyph Identifier*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

10.12.3 GLYPH INDEX MAP SPECIFICATION Processing

The processing of a GLYPH INDEX MAP SPECIFICATION shall consist of creating an Encoding Vector as defined in clause 25. For each index value *n* between 0 and the value of GLYPH INDEX MAP SIZE minus 1, the value of the element of the Encoding Vector whose index is *n* shall be generated as specified in 10.2.

The Glyph Index Map defined by the GLYPH INDEX MAP SPECIFICATION shall be the Glyph Index Map defined by the above Encoding Vector.

10.12.4 Standard Glyph Index Maps

Standard Glyph Index Maps are defined in 10.12.4. The Object Names for which *Public Object Identifier* values which identify Glyph Index Maps are defined by this International Standard, and the Glyph Index Maps identified by those values, are shown in table 15.

Table 15

Object Name	Glyph Index Map
GlyphIndexMap::Latin1Publishing	Latin 1 Publishing
GlyphIndexMap::Latin1PublishingA	Latin Publishing (A)
GlyphIndexMap::IR::nnnn	Algorithmic Glyph Index Map "*nnn*" , where "*nnn*" is either "0" or a sequence of decimal digits beginning with a non-zero digit

11 Document Production Instructions

Document Production Instructions perform two functions in the presentation of SPDL documents. The first function is to provide control over the aspects of document presentation which are outside of the scope of the document content notation. Examples of these are the selection of simplex or duplex printing modes and media specification and selection. The second function is to provide control over the aspects of document presentation which may vary from one instance of presentation to another. Examples of this are the number of copies to be printed and, again, media specification and selection. These two functions are not exclusive.

11.1 General

Each Document Production Instruction affects one or more presentation parameters. Each presentation parameter has a default value. The presentation parameter values in effect for the processing of a DOCUMENT are derived from the default values and any values specified in Supplementary DPI, and Document Production Instructions in the highest level BLOCK. The presentation parameter values in effect for the processing of subordinate BLOCKs are derived from the values of the presentation parameters for the most immediately superior BLOCK, Supplementary DPI, and Document Productions in the subordinate BLOCK. The derivation of the presentation parameter values used for processing each block is specified in clause 11 to clause 15.

NOTE 52 With the exception of the page-select presentation parameter, the default values of the presentation parameters are used only to initialize the presentation parameters at the beginning of document processing. In contrast, the default value of the page-select presentation parameter is used to initialize the page-select presentation parameter for each PAGESET. The processing of page-select presentation parameter for nested PAGESETs and the interaction of the page-select presentation parameter with the supplementary-page-select parameter are specified in clause 13.

For each Document Production Instruction, this International Standard specifies:

— the structure element which represents the Document Production Instruction;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642560 204

— the effect of the Document Production Instruction on one or more presentation parameters;

— the effect of the value of the affected presentation parameter or parameters on document presentation;

— the default value of the affected presentation parameter(s);

— the interaction of multiple occurrences of this Document Production Instruction within different BLOCKs of a document;

— the interaction of occurrences of this Document Production Instruction in Document DPI and in Supplementary DPI;

— fallback behavior for Presentation Processes which cannot achieve the specified effect.

11.2 Sources of Document Production Instructions

In order to affect those aspects of document presentation which are outside of the scope of the content notation, a method is provided for specifying Document Production Instructions in the document structure.

In order to affect those aspects of document presentation which may vary from one instance of presentation to another without changing the document for each instance of presentation, the user must be able to provide Document Production Instructions applicable to a particular instance of document presentation along with the document itself.

In any particular instance of presentation of an SPDL document, either or both of these sources may provide an empty set of Document Production Instructions.

11.2.1 Document Structure

Document Production Instructions which are specified in the document structure are called *Document DPI*.

Document DPI are specified by means of DPI DECLARATION structure elements. A Document Production Instruction is said to occur in a BLOCK if it is specified by a DPI DECLARATION in the PROLOGUE of that BLOCK.

11.2.2 Supplementary DPI

Document Production Instructions which are delivered with a document for a particular instance of document presentation are called *Supplementary DPI*. The method by which Supplementary DPI are made available to the Presentation Process is outside of the scope of this International Standard. An SPDL Presentation Process that has no capability of receiving Supplementary DPI shall be considered to have an empty set of Supplementary DPI for all instances of document presentation.

11.2.3 Additional DPI

In addition to those Document Production Instructions defined in this International Standard, the SPDL syntax allows for the inclusion of additional Document Production Instructions in Document DPI or Supplementary DPI. Such Document Production Instructions are called *Additional DPI*

Each Additional DPI consists of a name and a value. The name is a Public Object Identifier; the type of the value depends on the value of the name. While the specific syntax of Additional DPI and the effect of Additional DPI on the presentation of SPDL documents are outside of the scope of this International Standard, the inclusion of Additional DPI is specifically enabled.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642561 140

11.2.4 Reconciliation of Document Production Instructions

Since Document Production Instructions can occur in any BLOCK in an SPDL document, it is possible for Document Production Instructions in a superior BLOCK to be in conflict with Document Production Instructions in a subordinate BLOCK. Any such conflicts are reconciled by the Structure Processor.

The scope of a Document Production Instruction is the most immediately superior BLOCK of the PROLOGUE in which the DPI DECLARATION occurs. The effect of a Document Production Instruction may be modified or overridden by another Document Production Instruction.

Two or more Document Production Instructions have overlapping scope if one of them occurs within the scope of the other. The method of reconciliation of two Document Production instructions which have overlapping scope depends on the particular Document Production Instructions involved, and is specified as part of the semantics of each Document Production Instruction.

The scope of Supplementary DPI is the entire DOCUMENT. Therefore for any instance of presentation for which there is a non-null set of Supplementary DPI, there will be an overlap between the Supplementary DPI and the Document DPI. In this case the Structure Processor must reconcile the Supplementary DPI and the Document DPI. The method by which the Structure Processor reconciles Supplementary DPI with Document DPI depends on the Document Production Instructions involved, and is specified as part of the semantics of each Document Production Instruction.

11.3 Classes of Document Production Instructions

There are five classes of Document Production Instructions:

— *Pre-processing Document Production Instructions*, which affect the document production process prior to the rendering of the document page images;

— *Medium Document Production Instructions*, which specify the media to be used for document presentation and the selection of the medium to be used for the presentation of each page image;

— *Presentation Document Production Instructions*, which affect the process of rendering the page images and their placement on the media;

— *Finishing Document Production Instructions*, which affect the document production process subsequent to the rendering of the document page images;

— *Management Document Production Instructions*, which affect the management of the document production process.

There is an implied sequence for the application for Document Production Instructions.

— Pre-processing Document Production Instructions affect the process of document presentation prior to the rendering of the page images on the presentation medium. The result of the aggregate of the Pre-processing Instructions may be considered to be a new SPDL document. The new document is then presented using the original Presentation Instructions, Finishing Instructions, and Management Instructions.

  There are no pre-processing instructions defined by this International Standard.

— Medium Document Production Instructions apply to the result of the Pre-processing instructions.

— Presentation Document Production Instructions apply to the result of the Pre-processing instructions and the presentation of the resulting document using media selected by the Medium Document Production Instructions.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642562 087

— Finishing Document Production Instructions apply to the result of the process of rendering the document page images on the media. The sequence of application of Finishing Instructions may be significant.

— Management Document Production Instructions apply to all parts of the document production process, and are independent of the sequence of application of Pre-processing Instructions, Medium Instructions, Presentation Instructions, and Finishing Instructions.

11.4 Fallback

For some of the Document Production Instructions defined by this International Standard, the specified semantics can always be achieved. For these, no fallback specification is required. However, some of the Document Production Instructions specified in this International Standard require capabilities which need not be provided by every Presentation Process. In addition to the specification of operations for capable systems, this International Standard specifies the fallback behavior for these Document Production Instructions for systems in which those capabilities are not present. Where required, this fallback behavior is specified as part of the Document Production Instruction semantics.

There are some general classes of systems, such as soft copy devices, for which certain Document Production Instructions such as Finishing or Copies are generally inappropriate. In cases where a Document Production Instruction is inappropriate for document presentation by a particular system, the Document Production Instruction may be ignored.

11.5 Useful Structure elements

The following structure elements are used in several Document Production Instructions.

11.5.1 DIMENSION

The DIMENSION structure element represents a dimension. A DIMENSION structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one TOLERANCE structure element;

— one DIMENSION VALUE structure element.

The immediate subordinates shall occur in the above order.

11.5.1.1 TOLERANCE

A TOLERANCE structure element shall be a base structure element. The value of a TOLERANCE structure element shall be of type *Non-Negative Number*.

The value of a TOLERANCE structure element represents a dimensional tolerance.

11.5.1.2 DIMENSION VALUE

A DIMENSION VALUE structure element shall be either:

— a NAMED DIMENSION structure element;

— a NUMERIC DIMENSION structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

11.5.1.3 NAMED DIMENSION

A NAMED DIMENSION structure element shall be a base structure element. The value of a NAMED DIMENSION structure element shall be of type *Environment Identifier*.

The value of a NAMED DIMENSION structure element shall identify a value of type *Non-Negative Number*. There are no Public Object Identifier values for NAMED DIMENSION values defined by this International Standard.

11.5.1.4 NUMERIC DIMENSION

A NUMERIC DIMENSION VALUE structure element shall be a base structure element. The value of a NUMERIC DIMENSION VALUE structure element shall be of type *Non-Negative Number*.

11.5.2 XYDIMENSIONS

An XYDIMENSIONS structure element represents the dimensions of a rectangular sheet medium or a location in the Reference Coordinate System. An XYDIMENSIONS structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one TOLERANCE structure element;

— one XYDIMENSIONS VALUE structure element.

The immediate subordinates shall occur in the above order.

11.5.2.1 XYDIMENSIONS VALUE

An XYDIMENSIONS VALUE structure element shall be one of:

— a NAMED XYDIMENSIONS structure element;

— a NUMERIC XYDIMENSIONS structure element.

11.5.2.2 NAMED XYDIMENSIONS

A NAMED XYDIMENSIONS structure element shall be a base structure element. The value of a NAMED XYDIMENSIONS structure element shall be of type *Environment Identifier*.

The value of a NAMED XYDIMENSIONS structure element shall identify an ordered pair of values of type *Non-Negative Number*. The value most commonly represents the dimensions of a medium. Public Object Identifier values for NAMED XYDIMENSIONS are defined by ISO/IEC 10175 as standard values for the attribute *medium-size*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

11.5.2.3 NUMERIC XYDIMENSIONS

A NUMERIC XYDIMENSIONS structure element shall be a composite structure element which shall have the following immediate subordinates:

— one X-DIMENSION structure element;

— one Y-DIMENSION structure element;

The immediate subordinates shall occur in the above order.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642564 95T

11.5.2.4 X-DIMENSION

An X-DIMENSION structure element shall be a base structure element. The value of an X-DIMENSION structure element shall be of type *Non-Negative Number*.

11.5.2.5 Y-DIMENSION

A Y-DIMENSION structure element shall be a base structure element. The value of a Y-DIMENSION structure element shall be of type *Non-Negative Number*.

11.5.3 HEAD LOCATIONS

A HEAD LOCATIONS structure element represents a sequence of head locations for a finishing operation. A HEAD LOCATIONS structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one TOLERANCE structure element;

— one HEAD LOCATIONS VALUE structure element.

The immediate subordinates shall occur in the above order.

11.5.3.1 HEAD LOCATIONS VALUE

A HEAD LOCATIONS VALUE structure element shall be one of:

— a NAMED HEAD LOCATIONS structure element;

— a NUMERIC HEAD LOCATIONS structure element.

11.5.3.2 NAMED HEAD LOCATIONS

A NAMED HEAD LOCATIONS structure element shall be a base structure element. The value of a NAMED HEAD LOCATIONS structure element shall be of type *Environment Identifier*.

The value of a NAMED HEAD LOCATIONS structure element shall identify a sequence of values $x_1, \ldots, x_n$ of type *Non-Negative Number* such that for each value of i, $1 \sigma i \sigma n-1$, $x_i \sigma x_i + 1$. There are no Public Object Identifier values for NAMED HEAD LOCATIONS values defined by this International Standard.

11.5.3.3 NUMERIC HEAD LOCATIONS

A NUMERIC HEAD LOCATIONS structure element shall be a composite structure element which shall have the following immediate subordinates:

— one or more HEAD LOCATION structure elements.

The values of the immediately subordinate HEAD LOCATION structure elements shall consist of a sequence of values $x_1, \ldots, x_n$ such that for each value of i, $1 \sigma i \sigma n-1$, $x_i \sigma x_i + 1$.

11.5.3.4 HEAD LOCATION

A HEAD LOCATION structure element shall be a base structure element. The value a HEAD LOCATION structure elements shall be of type *Non-Negative Number*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.5.4 EDGE

An EDGE structure element identifies a distinguished edge of a medium or of the assembled document. An EDGE structure element shall be a base structure element. The type of the value of an EDGE structure element shall be an Enumerated type. The values of this type and the edges identified shall be as shown in table 16.

Table 16

Value	Edge
bottom-edge	the edge coincident with the x-axis of the Reference Coordinate System
right-edge	the edge parallel to the y-axis of the Reference Coordinate System and opposite the edge coincident with the y-axis
top-edge	the edge parallel to the x-axis of the Reference Coordinate System and opposite the edge coincident with the x-axis
left-edge	the edge of the medium coincident with the y-axis of the Reference Coordinate System

11.6 DPI DECLARATION

A DPI DECLARATION consists of a set of zero or more Document Production Instructions. The DPI DECLARATION syntax supports not only the Document Production Instructions which are defined by this International Standard but also Additional DPI.

A DPI DECLARATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one MEDIUM DPI structure element;

— zero or one MEDIUM SELECT DPI structure element;

— zero or one CURRENT MEDIUM DPI structure element;

— zero or one COPIES DPI structure element;

— zero or one PAGE SELECT DPI structure element;

— zero or one SUPPLEMENTARY PAGE SELECT DPI structure element;

— zero or one SIDES DPI structure element;

— zero or one PLEX DPI structure element;

— zero or one X-IMAGE SHIFT DPI structure element;

— zero or one Y-IMAGE SHIFT DPI structure element;

— zero or one CURRENT SIDE DPI structure element;

— zero or one FINISHING DPI structure element;

— zero or one COLORANT SET DPI structure element;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642566 722

— zero or one DOCUMENT START MESSAGE DPI structure element;

— zero or one AUXILIARY PAGE TYPE DPI structure element;

— zero or one DOCUMENT COMMENT DPI structure element;

— zero or one TIMEOUT DPI structure element;

— zero or one ABORT POLICY DPI structure element;

— zero or one OUTPUT BIN DPI structure element;

— zero or one OUTPUT SPECIFICATION DPI structure element;

— zero or one DOCUMENT END MESSAGE DPI structure element;

— zero or more ADDITIONAL DPI structure elements.

The immediate subordinates may occur in any order.

11.7 Additional Document Production Instructions

Each Additional DPI consists of a name and a value. The general syntax for Additional DPI is specified in this clause. The specific syntax and the semantics of an Additional DPI are specified by the owner of the name. While the specific syntax of Additional DPI and the effect of Additional DPI on the presentation of SPDL documents are outside of the scope of this International Standard, the inclusion of Additional DPI is specifically enabled.

Additional Document Production Instructions may be any of:

— *Pre-processing Document Production Instruction*;

— *Presentation Document Production Instruction*;

— *Finishing Document Production Instruction*;

— *Management Document Production Instruction*.

11.7.1 ADDITIONAL DPI

An ADDITIONAL DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

— one DPI NAME structure element;

— zero or one DPI VALUE structure elements.

The immediate subordinates shall occur in the specified order.

11.7.2 DPI NAME

A DPI NAME structure element shall be a base structure element. The value of a DPI NAME structure element shall be of type *Public Object Identifier*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642567 669

11.7.3 DPI VALUE

A DPI VALUE structure element may be either a base structure element or a composite structure element. The DPI VALUE structure element shall depend on the value of the peer DPI NAME structure element.

11.7.4 ADDITIONAL DPI Semantics

The specification of the semantics of an ADDITIONAL DPI shall be the responsibility of the owner of the Public Object Identifier which is the value of the DPI NAME. The effect of an Additional DPI on the presentation of an SPDL document shall be either:

— an effect defined by the semantics of the Additional DPI;

— that the Additional DPI is ignored.

The definition of the syntax of the DPI VALUE corresponding to any particular DPI NAME value, and the effect of ADDITIONAL DPI on the presentation of SPDL documents are outside of the scope of this International Standard.

12 Medium Document Production Instructions

Medium Document Production Instructions specify the media to be used for document presentation and the selection of the medium to be used for the presentation of each page image. The result of the Medium Declaration and Selection Process is the association of a medium with each ordinal page number. The placement of page images on instances of the media is specified in 16.7.

For web fed printers or other printers using continuous media, an instance of the medium is considered to be that portion of the medium on which a single page image can be placed. Such instances of the media would be separated by the Presentation Process after the page images are placed on them. Presentation Processes such as softcopy devices may place page images on virtual presentation surfaces which can then be viewed on the physical presentation surface. In such a system each virtual presentation surface is treated as a separate instance of the medium. The mapping of these virtual presentation surfaces to the physical display surface and the ability of the system to preserve a virtual presentation surface for future viewing are system dependent.

The specification of the media which are to be used for document presentation is a two step process.

a) Each medium which may be used for the presentation of one or more pages is first declared by a Medium Declaration.

b) The medium which is to be used for the presentation of each PAGE is selected by the Medium Select or Current Medium Document Production Instructions.

The presentation parameters which are affected by the Medium Document Production Instructions are shown in table 17.

Table 17

Parameter	Affects
medium-list	media available for document presentation
medium-select	selection of media for pages in the document by a Medium Select Document Production Instruction

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Parameter	Affects
current-medium	selection of media for pages in a BLOCK by a Current Medium Document Production Instruction

12.1 Medium Declaration

Each medium which may be used for the presentation of one or more pages is first declared by a Medium Declaration.

12.1.1 Medium Specification

A medium may be specified by a medium name, by a set of medium properties, or by a combination of medium name and medium properties.

For some of the medium properties, the set of values for which a meaning is specified may not include all of the possible values of the assigned type. A MEDIUM PROPERTIES structure element may specify a value for a medium property which is other than one of those for which a meaning is defined in 12.5 so long as the value is of the appropriate type. The effect of such a medium property is system dependent.

A MEDIUM PROPERTIES structure element may also specify medium properties in addition to those defined by this International Standard. The effect of such a medium property is system dependent.

12.1.2 Selection of Physical Media

The primary means of medium identification is the MEDIUM NAME. The MEDIUM NAME may identify either a specific medium (such as a specific product) or a type of medium (such as ISO A4 paper). If the MEDIUM NAME identifies a specific medium, any additional properties are used for medium substitution only. If the MEDIUM NAME identifies a type of medium, other medium properties may be used to select from among the media of that type which are available to the Presentation Process.

NOTE 53 An example of a MEDIUM NAME which identifies a type of medium is the Public Object Identifier which identifies ISO A4 paper. Additional properties which might be used with this MEDIUM NAME value include medium weight and medium color.

If a MEDIUM SPECIFICATION does not include MEDIUM NAME value, then a medium is selected using the medium properties. The medium properties may be specified explicitly by a MEDIUM PROPERTIES structure element, implied by the medium name or medium type properties, or default values may be assumed. The medium properties which are defined by this International Standard are shown in table 18.

Table 18

Property	Meaning
medium-size	the size of the medium
medium-color	the color of the medium
medium-weight	the weight of the medium
medium-type	the type of the medium
medium-tooth	the tooth of the medium
medium-grain	the grain of the medium

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Property	Meaning
medium-finish-edge	an edge of the medium which has a distinguishing property

The default values for these properties are specified by this clause.

The method of assigning a medium to a MEDIUM SPECIFICATION which does not specify a MEDIUM NAME is system dependent.

The ability of systems to select media from an available repertoire is system dependent. There is no requirement for a system to have any particular repertoire of media available, or to be able to select media from a repertoire of available media. This International Standard specifies only how a system which can select from a repertoire of available media assigns a particular medium to a MEDIUM SPECIFICATION.

NOTE 54 A system may assign a single default medium, or a medium which depends on the current state of the system (e.g. whatever medium is in a particular input tray), to every MEDIUM SPECIFICATION.

12.1.3 Medium-list presentation parameter

The list of media which may be used for the presentation of a particular page of the document is specified by the current value of the *medium-list* presentation parameter. The *medium-list* presentation parameter is a list of *name/medium* pairs for which:

— the *name* element is a Name assigned by a MEDIUM DECLARATION;

— the *medium* element represents the physical medium which best satisfies the MEDIUM SPECIFICATION of that MEDIUM DECLARATION.

The default value of the *medium-list* presentation parameter is a null list. Each MEDIUM DECLARATION causes a *name/medium* pair to be added to the *medium-list* for the scope of the Medium Document Production Instruction.

12.1.4 Default medium

In addition to the media which are identified by Medium Declarations, there is a default medium for each instance of presentation. The means by which the default medium is established is outside of the scope of this International Standard.

NOTE 55 For a Presentation Process which is part of a ISO/IEC 10175 print service, the default medium may be set by the value of the default-medium Print Operation Parameter.

12.2 Association of Media with Ordinal Page Numbers

The result of the Medium Declaration and Selection Process is the association of a medium with each ordinal page number. The selection of the medium to be associated with each ordinal page number is determined by two presentation parameters:

— the *medium-select* presentation parameter;

— the *current-medium* presentation parameter.

The *medium-select* presentation parameter is set by the Medium Select Document Production Instruction, which is only valid in Supplementary DPI. The *current-medium* presentation parameter is set by the Current Medium Document Production Instruction, which is only valid in Document DPI.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642570 153

12.2.1 Medium Selection by Medium Select

The *medium-select* presentation parameter is used to establish the association of a physical medium with each ordinal page number whenever there is a Medium Select Document Production Instruction for a particular instance of presentation. The value of the *medium-select* presentation parameter is a sequence of Names, which specifies the sequence of media to be used for the presentation of the pages of the entire document.

If there are *n* PAGEs in the document, then for 1σισn the medium associated with the ordinal page number *i* is determined from the *medium-select*, *current-medium* and *medium-list* presentation parameters as follows:

a) if *medium-select(i)* is ".notdef", the medium associated with the ordinal page number *i* is determined by the *current-medium* presentation parameter as specified below;

b) if *medium-select(i)* is other than ".notdef" and there is an element of *medium-list* which binds the value of *medium-select(i)* to a physical medium, the medium associated with the ordinal page number *i* is the medium which is bound to the value of *medium-select(i)*;

c) if *medium-select(i)* is other than ".notdef" but there is no element of *medium-list* which binds the value of *medium-select(i)* to a physical medium, a structure warning occurs and the medium associated with the ordinal page number *i* is determined by the *current-medium* presentation parameter.

12.2.2 Medium Selection by Current Medium

The *current-medium* presentation parameter is used to establish the association of a physical medium with each ordinal page number whenever there is no Medium Select Document Production Instruction for a particular instance of presentation or whenever the value of the *medium-select* presentation parameter fails to specify a medium. The value of the *current-medium* presentation parameter is a single Name, which specifies the medium to be used for the presentation of the current BLOCK.

If there are *n* PAGEs in the document and if the value of the *current-medium* presentation parameter during the processing of the PAGE which is assigned the ordinal page number *i* is *current-medium(i)*, then for 1σισn the medium associated with the ordinal page number *i* is determined from the *current-medium* and *medium-list* presentation parameter as follows

a) if *current-medium(i)* is ".notdef", the medium associated with the ordinal page number *i* is the default medium:

b) if *current-medium(i)* is other than ".notdef" and there is an element of *medium-list* which binds the value of *current-medium(i)* to a physical medium, the medium associated with the ordinal page number *i* is the medium which is bound to the value of *current-medium(i)*;

c) if *current-medium(i)* is other than ".notdef" but there is no element of *medium-list* which binds the value of *current-medium(i)* to a physical medium, a structure warning occurs and the medium associated with the ordinal page number *i* is the default medium

12.3 Medium Document Production Instruction

The Medium Document Production Instruction identifies media which may be used in document processing.

12.3.1 MEDIUM DPI

The Medium Document Production Instruction shall be represented by a MEDIUM DPI structure element. A MEDIUM DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— one or more MEDIUM DECLARATION structure elements;

12.3.1.1 MEDIUM DECLARATION

A MEDIUM DECLARATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one MEDIUM IDENTIFIER structure element;

— one MEDIUM SPECIFICATION structure element.

The immediate subordinates shall occur in the above order.

12.3.1.2 MEDIUM IDENTIFIER

A MEDIUM IDENTIFIER structure element shall be a base structure element The value of a MEDIUM IDENTIFIER structure element shall be of type *Name*.

12.3.2 MEDIUM Document Production Instruction Processing

The effect of the Medium Document Production Instruction shall be to append or replace an element in the *medium-list* presentation parameter for each MEDIUM DECLARATION which is immediately subordinate to the MEDIUM DPI structure element. For each such MEDIUM DECLARATION, the effect shall be as follows.

a) If there is no *name/medium* pair in the current *medium-list* for which the name is the same as the value of MEDIUM IDENTIFIER in the MEDIUM DECLARATION, the effect shall be to append a name/medium pair to the *medium-list* for which

   — the *name* is the same as the value of the MEDIUM IDENTIFIER in the MEDIUM DECLARATION;

   — the *medium* is the physical medium in the repertoire of the Presentation Process which best satisfies the MEDIUM DECLARATION;

   for the scope of the Medium Document Production Instruction.

b) If there is a *name/medium* pair in the current *medium-list* for which the *name* is the same as the value of MEDIUM IDENTIFIER in the MEDIUM DECLARATION, the effect shall be to replace that *name/medium* pair with a new *name/medium* pair for which

   — the *name* is the same as the value of the MEDIUM IDENTIFIER in the MEDIUM DECLARATION;

   — the *medium* is the physical medium in the repertoire of the Presentation Process which best satisfies the MEDIUM DECLARATION;

   for the scope of the Medium Document Production Instruction.

NOTE 56 If two MEDIUM DECLARATION structure elements in different BLOCKs have the same subordinate MEDIUM IDENTIFIER value, the MEDIUM DECLARATION in the subordinate BLOCK takes precedence for the scope of that DPI DECLARATION.

12.3.3 Medium Document Production Instruction Reconciliation

A Medium Document Production Instruction is meaningful in any PAGE or PAGESET or in Supplementary DPI. Any Medium Document Production Instruction which occurs elsewhere is ignored.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642572 T26

If the value of the MEDIUM IDENTIFIER of a MEDIUM DECLARATION of a Supplementary Medium Document Production Instruction has the same value as the MEDIUM IDENTIFIER of the MEDIUM DECLARATION of a Document Medium Document Production Instruction, the MEDIUM DECLARATION in the Supplementary DPI shall take precedence.

12.3.4 Medium Document Production Instruction Fallback

There is no fallback required for the Medium Document Production Instruction.

12.4 MEDIUM SPECIFICATION

A MEDIUM SPECIFICATION structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one MEDIUM NAME structure element;

— zero or one MEDIUM MESSAGE structure element;

— zero or one MEDIUM PROPERTIES structure element.

At least one of the immediate subordinates shall be present. The immediate subordinates may occur in any order.

12.4.1 MEDIUM NAME

A MEDIUM NAME structure element shall be a base structure element. The value of a MEDIUM NAME structure element shall be of type *Environment Identifier*.

The value of a MEDIUM NAME structure element shall identify a medium as a whole. Public Object Identifier values for MEDIUM NAME are defined by ISO/IEC 10175 as standard values for the attribute *medium*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

12.4.2 MEDIUM MESSAGE

A MEDIUM MESSAGE structure element shall be a base structure element. The value of a MEDIUM MESSAGE structure element shall be of type *Printable String*.

The MEDIUM MESSAGE specifies a message which may describe the medium, the suggested fallback action, or provide other information. It is intended that this message be delivered to an operator prior to and as close as possible to the printing of the document.

A MEDIUM MESSAGE shall have no effect on medium selection, but may effect printing management. The effect of a MEDIUM MESSAGE on printing management shall be system dependent.

12.4.3 MEDIUM SPECIFICATION Processing

MEDIUM SPECIFICATION processing consists of identifying a physical medium in the environment of the Presentation Process on the basis of the medium properties.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642573 962

12.5 MEDIUM PROPERTIES

Medium properties are specified by a MEDIUM PROPERTIES structure element. A MEDIUM PROPERTIES structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one MEDIUM SIZE structure element;

— zero or one MEDIUM COLOR structure element;

— zero or one MEDIUM WEIGHT structure element;

— zero or one MEDIUM TYPE structure element;

— zero or one MEDIUM TOOTH structure element;

— zero or one MEDIUM GRAIN structure element;

— zero or one MEDIUM PRE-PUNCH structure element;

— zero or one MEDIUM MULTI-PART structure element;

— zero or one MEDIUM FINISHED EDGES structure element;

— zero or one MEDIUM LABELS structure element;

— zero or more ADDITIONAL PROPERTY structure elements.

The immediate subordinates may occur in any order.

12.5.1 MEDIUM SIZE

A MEDIUM SIZE structure element shall be an XYDIMENSIONS structure element.

The MEDIUM SIZE structure element specifies a value for the *medium-size* medium property. The value of the *medium-size* medium property specifies the size of a rectangular medium and the relationship of the Reference Coordinate System to an instance of the medium. The value of the *medium-size* property shall be an ordered pair of Numbers. A *medium-size* medium property value of (*x*, *y*) specifies that the size of the medium is *x* mm by *y* mm, and that the Reference Coordinate System is associated with each instance of the medium in such a way that:

— one of the edges of length *x* mm corresponds to the positive *x*-axis of the Reference Coordinate system;

— one of the edges of length *y* mm corresponds to the positive *y*-axis of the Reference Coordinate system.

Public Object Identifier values for *medium-size* are defined by ISO/IEC 10175 as standard values for the attribute *medium-size*. The Public Object Identifier values which are defined by ISO/IEC 10175 for the attribute *medium-size* and the meaning of those values are provided in annex F (informative).

The default value of the *medium-size* medium property shall be system dependent

NOTE 57 Commonly, the default will be either (216, 279) specifying 8 inch by 11 inch paper, or (210, 297), specifying ISO A4 paper.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642574 8T9

12.5.2 MEDIUM COLOR

A MEDIUM COLOR structure element shall be a base structure element. The value of a MEDIUM COLOR structure element shall be of type *Environment Identifier*.

A MEDIUM COLOR structure element specifies the value of the *medium-color* medium property. The value of the *medium-color* medium property specifies the color of the medium. Public Object Identifier values for *medium-color* are defined by ISO/IEC 10175 as standard values for the attribute *medium-colour*. The Public Object Identifier values which are defined by ISO/IEC 10175 for the attribute *medium-colour* and the meaning of those values are provided in annex F (informative).

The default value of the *medium-color* medium property shall be *dpi::medium::colour::white*.

12.5.3 MEDIUM WEIGHT

A MEDIUM WEIGHT structure element shall be a composite structure element which shall have the following immediate subordinates:

— one MEDIUM WEIGHT VALUE structure element;

— zero or one MEDIUM WEIGHT TOLERANCE structure element.

The immediate subordinates may occur in any order.

12.5.3.1 MEDIUM WEIGHT VALUE

A MEDIUM WEIGHT VALUE structure element shall be a base structure element. The value of a MEDIUM WEIGHT VALUE structure element shall be of type *Cardinal*.

12.5.3.2 MEDIUM WEIGHT TOLERANCE

A MEDIUM WEIGHT TOLERANCE structure element shall be a base structure element. The value of a MEDIUM WEIGHT TOLERANCE structure element shall be of type *Cardinal*

12.5.3.3 MEDIUM WEIGHT Processing

A MEDIUM WEIGHT structure element specifies the value of the *medium-weight* medium property.

The *medium-weight* medium property specifies the weight of the medium in grams per square meter (gm/m^2), rounded to the nearest conventional value.

The default value of the *medium-weight* medium property shall be system dependent.

NOTE 58 Commonly, the default will be 75, which is equivalent to paper which is identified as "substance 20", or "20 lb".

12.5.4 MEDIUM TYPE

A MEDIUM TYPE structure element shall be a base structure element. The value of a MEDIUM TYPE structure element shall be of type *Environment Identifier*.

A MEDIUM TYPE structure element specifies the value of the *medium-type* medium property. The value of the *medium-type* medium property identifies the general type of of the medium, such as paper, adhesive backed medium, continuous form paper, or envelopes. Public Object Identifier values for MEDIUM TYPE are defined by ISO/IEC 10175 as standard values for the attribute *medium-type*. The Public Object Identifier values which

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642575 735

are defined by ISO/IEC 10175 for the attribute *medium-type* are and the meaning of those values are provided in annex F (informative).

The default value of the *medium-type* medium property shall be *dpi::medium::type::stationery*.

12.5.5 MEDIUM TOOTH

A MEDIUM TOOTH structure element shall be a base structure element. The value of a MEDIUM TOOTH structure element shall be of type *Environment Identifier*.

A MEDIUM TOOTH structure element specifies the value of the *medium-tooth* medium property. The value of the *medium-tooth* medium property specifies the tooth of the medium. Public Object Identifier values for MEDIUM TOOTH are defined by ISO/IEC 10175 as standard values for the attribute *medium-tooth*. The Public Object Identifier values which are defined by ISO/IEC 10175 for the attribute *medium-tooth* and the meaning of those values are provided in annex F (informative).

The default value of the *medium-tooth* medium property shall be system dependent.

12.5.6 MEDIUM GRAIN

A MEDIUM GRAIN structure element shall be a base structure element. The value of a MEDIUM GRAIN structure element shall be of type *Environment Identifier*.

A MEDIUM GRAIN structure element specifies the value of the *medium-grain* medium property. The value of a MEDIUM GRAIN structure element identifies the grain of the medium. Public Object Identifier values for MEDIUM GRAIN are defined by ISO/IEC 10175 as standard values for the *medium-grain* attribute. The Public Object Identifier values defined by ISO/IEC 10175 for the *medium-grain* attribute and the meaning of those values are described in annex F (informative).

The default value of the *medium-grain* medium property shall be system dependent.

12.5.7 MEDIUM PRE-PUNCH

A MEDIUM PRE-PUNCH structure element shall be a composite structure element which shall have the following immediate subordinates:

— one PUNCH COUNT structure element;

— zero or one PUNCH EDGE structure element;

— zero or one PUNCH DIAMETER structure element;

— zero or one PUNCH OFFSET structure element;

— zero or one PUNCH LOCATIONS structure element.

The immediate subordinates may occur in any order.

12.5.7.1 PUNCH COUNT

A PUNCH COUNT structure element shall be a base structure element. The value of a PUNCH COUNT structure element shall be of type *Cardinal*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

12.5.7.2 PUNCH EDGE

A PUNCH EDGE structure element shall be an EDGE structure element.

12.5.7.3 PUNCH DIAMETER

A PUNCH DIAMETER structure element shall be a DIMENSION structure element.

12.5.7.4 PUNCH OFFSET

A PUNCH OFFSET structure element shall be a DIMENSION structure element.

12.5.7.5 PUNCH LOCATIONS

A PUNCH LOCATIONS structure element shall be a HEAD LOCATIONS structure element. The number of Non-Negative Numbers shall be the same as the value of PUNCH COUNT.

12.5.7.6 MEDIUM PRE-PUNCH Processing

A MEDIUM PRE-PUNCH structure element specifies the value of the *medium-pre-punch* medium property. The *medium-pre-punch* medium property specifies the selection of a medium with one or more pre-punched or pre-drilled holes. The holes are inline as in the case of a sheet used for ring binding.

A MEDIUM PRE-PUNCH structure element for which the value of PUNCH COUNT is non-zero specifies a value for the *medium-pre-punch* medium property for which the punch count is the value of the PUNCH COUNT structure element and for which the punch parameters are as follows.

- The punch edge identifies the nearest edge parallel to the pre-punched holes. The punch edge is identified by the value of the PUNCH EDGE structure element if present, or is the *left edge* if no PUNCH EDGE structure element is present.

- The punch offset is the offset of the centerline of the sequence of pre-punched holes from the punch edge in millimeters. The punch offset is the value of the PUNCH OFFSET structure element if present, or a system dependent value which may depend on the value of punch count if no PUNCH OFFSET structure element is present,

- The punch diameter is the diameter of the pre-punched holes in millimeters. The punch diameter is the value of the PUNCH DIAMETER structure element if present, or a system dependent value which may depend on the value of punch count if no PUNCH DIAMETER structure element is present;

- Punch locations are the locations of the punched holes relative to a coordinate system whose y-axis lies along the punch edge and whose x-axis lies along an edge perpendicular to the punch edge, so that the medium lies in the quadrant containing positive x and y values. The punch locations consist of a sequence of Non-Negative Numbers which specify the locations of the pre-punched or pre-drilled holes as offsets from the x-axis, in millimeters. The number of Non-Negative Numbers is the same as the value of punch count. Punch locations are specified by the value of the PUNCH LOCATIONS structure element if present, or a system dependent value which may depend on the value of punch count if no PUNCH LOCATIONS structure element is present.

The default value of the *medium-pre-punch* medium property shall have punch count equal to zero. The *medium-pre-punch* medium property is optional. If no value is specified, the medium shall have no pre-punched or pre-drilled holes.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

12.5.8 MEDIUM MULTI-PART

A MEDIUM MULTI-PART structure element shall be a composite structure element which shall have the following immediate subordinates:

— one SEQUENCE NUMBER structure element;

— one SEQUENCE LENGTH structure element.

The immediate subordinates may occur in any order.

12.5.8.1 SEQUENCE NUMBER

A SEQUENCE NUMBER structure element shall be a base structure element. The value of a SEQUENCE NUMBER structure element shall be of type *Cardinal*.

12.5.8.2 SEQUENCE LENGTH

A SEQUENCE LENGTH structure element shall be a base structure element. The value of a SEQUENCE LENGTH structure element shall be of type *Cardinal*.

12.5.8.3 MEDIUM MULTI-PART Processing

A MEDIUM MULTI-PART structure element specifies the value of the *medium-multi-part* medium property. The *medium-multi-part* medium property is meaningful only when the value of MEDIUM TYPE specifies a multi-part medium.

The *medium-multi-part* medium property may specify either

— a medium which is the nth of m media of a particular multi-part type which are managed as separate media;

— collated sequentially accessed media of a particular multi-part type which are managed as a single medium.

An example of a medium of the first type is fifth-cut tab stock with all of the tabs cut in the first (or second, or third) position. An example of a medium of the second type is fifth-cut tab stock which is packaged in collated form (a sheet of tab stock with the tab cut in the first position, followed by a sheet of tab stock with the tab cut in the second position, etc.).

A value of the *medium-multi-part* medium property shall consist of:

— one sequence number;

— one sequence length.

A MEDIUM MULTI-PART structure element specifies a value for the *medium-multi-part* medium property for which:

— sequence number is the value of the SEQUENCE NUMBER structure element;

— sequence length is the value of the SEQUENCE LENGTH structure element.

A *medium-multi-part* medium property for which the value of sequence number is n and the value of sequence length is m specifies that the medium is the nth of m media of a particular multi-part type which are managed as separate media. A *medium-multi-part* medium property for which the value of sequence number is zero specifies that the medium consists of collated sequentially accessed media of a particular type. In this case the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642578 444

value of the sequence length helps to identify the medium (e.g. to distinguish third-cut tab stock from fifth-cut tab stock).

NOTES

59 The use of a MEDIUM MULTI-PART property for which the value of SEQUENCE NUMBER is non-zero implies independent access to each of the parts. In general, this requires the printer to use a separate tray for each part.

60 If collated sequentially accessed media are used, the output for multiple copies may depend on how copies are made. For example, a system which uses a sorter will generally produce different results than one which images all pages of one copy followed by all pages of the next copy.

The *medium-multi-part* medium property is optional.

12.5.9 MEDIUM FINISHED EDGES

A MEDIUM FINISHED EDGES structure element shall be a composite structure element which shall have the following immediate subordinates:

— one to four FINISHED EDGE structure elements.

12.5.9.1 FINISHED EDGE

A FINISHED EDGE structure element shall be a composite structure element which shall have the following immediate subordinates:

— one EDGE IDENTIFIER structure element;

— zero or one EDGE FINISH TYPE structure element.

The immediate subordinates may occur in any order.

12.5.9.2 EDGE IDENTIFIER

An EDGE IDENTIFIER structure element shall be an EDGE structure element.

12.5.9.3 EDGE FINISH TYPE

An EDGE FINISH TYPE structure element shall be a base structure element. The value of an EDGE FINISH TYPE structure element shall be of type *Environment Identifier*. There are no Object Names for which Public Object Identifier values for EDGE FINISH TYPE are defined by this International Standard.

12.5.9.4 MEDIUM FINISHED EDGES Processing

A MEDIUM FINISHED EDGE structure element specifies the value of the *medium-finished-edges* medium property. The *medium-finished-edges* medium property specifies one or more finished edges of the medium and the ways in which those edges have been finished. The *medium-finished-edges* medium property may be used for the purposes of identifying and physically orienting certain types of media (such as a medium with a deckled edge).

12.5.10 MEDIUM LABELS

A MEDIUM LABELS structure element shall be a composite structure element which shall have the following immediate subordinates:

— one PER COLUMN structure element;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642579 380

— one PER ROW structure element.

The immediate subordinates may occur in any order.

12.5.10.1 PER COLUMN

A PER COLUMN structure element shall be a base structure element. The value of a PER COLUMN structure element shall be of type *Cardinal*.

12.5.10.2 PER ROW

A PER ROW structure element shall be a base structure element. The value of a PER ROW structure element shall be of type *Cardinal*.

12.5.10.3 MEDIUM LABELS Processing

A MEDIUM LABELS structure element specifies the value of the *medium-labels* medium property. The *medium-labels* medium property is meaningful only when the value of MEDIUM TYPE specifies a type of label stock.

The *medium-labels* medium property specifies the layout of pre-cut labels on a type of medium commonly referred to as label stock. A *medium-labels* value for which *labels-per-column* is *n* and *labels-per-row* is *m* specifies a medium which consists of label stock which contains $n \times m$ pre-cut labels per sheet: *n* pre-cut labels per column and *m* pre-cut labels per row.

A MEDIUM LABELS structure element shall specify a value for the *medium-labels* medium property for which:

— *labels-per-column* is the value of the PER COLUMN structure element;

— *labels-per-row* the value of the PER ROW structure element.

12.5.11 ADDITIONAL PROPERTY

The MEDIUM PROPERTIES structure element may specify additional medium properties by means of one or more ADDITIONAL PROPERTY structure elements. The effect of such ADDITIONAL PROPERTY structure elements is system dependent.

An ADDITIONAL PROPERTY structure element shall be a composite structure element which shall have the following immediate subordinates:

— one PROPERTY NAME structure element;

— zero or one PROPERTY VALUE structure elements.

The immediate subordinates may occur in any order.

12.5.11.1 PROPERTY NAME

A PROPERTY NAME structure element shall be a base structure element. The value of a PROPERTY NAME structure element shall be of type *Public Object Identifier*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642580 0T2

12.5.11.2 PROPERTY VALUE

A PROPERTY VALUE structure element may be either a base structure element or a composite structure element. The PROPERTY VALUE structure element shall depend on the value of the PROPERTY NAME structure element.

12.6 Medium Select Document Production Instruction

The Medium Select Document Production Instruction shall be represented by a MEDIUM SELECT DPI structure element.

12.6.1 MEDIUM SELECT DPI

A MEDIUM SELECT DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or more MEDIUM SELECTION structure elements.

12.6.1.1 MEDIUM SELECTION

A MEDIUM SELECTION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one START PAGE structure element;

— one END PAGE structure element;

— one MEDIUM IDENTIFIER structure element.

These structure elements may occur in any order.

For each MEDIUM SELECT structure element, the value of the immediately subordinate END PAGE structure element shall be greater than or equal to the value of START PAGE structure element, and the value of the immediately subordinate START PAGE structure element shall be larger than the value of the END PAGE structure element which is subordinate to the immediately preceding MEDIUM SELECTION (if any).

12.6.1.2 START PAGE

A START PAGE structure element shall be a PAGE IDENTIFIER structure element

12.6.1.3 END PAGE

An END PAGE structure element shall be a PAGE IDENTIFIER structure element.

12.6.1.4 PAGE IDENTIFIER

A PAGE IDENTIFIER structure element shall be a base structure element. The value of a PAGE IDENTIFIER structure element shall be of type *Positive Integer*.

12.6.1.5 MEDIUM IDENTIFIER

A MEDIUM IDENTIFIER structure element shall be a base structure element. The value of a MEDIUM IDENTIFIER structure element shall be of type *Environment Identifier*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

12.6.2 Medium Select Document Production Instruction Processing

The effect of the MEDIUM SELECT DPI shall be to set the value of the *medium-select* presentation parameter. If the value of the START PAGE which is subordinate to the nth MEDIUM SELECTION is $start\text{-}page_n$, the value of the END PAGE whch is subordinate to the nth MEDIUM SELECTION is $end\text{-}page_n$, and the value of the MEDIUM IDENTIFIER which is subordinate to the nth MEDIUM SELECTION is $medium\text{-}name_n$, then the value of *medium-select* shall be obtained by:

a) setting the initial value of the *medium-select* presentation parameter to

$$medium\text{-}select(i) = \text{".notdef"}$$

for all values of i;

b) for each MEDIUM SELECTION, setting

$$medium\text{-}select(i) = Medium\text{-}Name_n$$

for all values of i for which $start\text{-}page_n \sigma i \sigma end\text{-}page_n$.

If there are no subordinate MEDIUM SELECTION structure elements, the value of the *medium-select* presentation parameter shall be $medium\text{-}select(i) =$ ".notdef" for all values of i.

12.6.3 Medium Select Document Production Instruction Reconciliation

The Medium Select Document Production Instruction is meaningful only in a DOCUMENT or in Supplementary DPI. Any Medium Select Document Production Instruction which occurs in a BLOCK which is not a DOCUMENT shall be ignored.

12.6.4 Medium Select Document Production Instruction Fallback

There is no fallback required for the Medium Select Document Production Instruction.

12.7 Current Medium Document Production Instruction

The Current Medium Document Production Instruction specifies the medium to be used for the presentation of the pages of a BLOCK.

12.7.1 CURRENT MEDIUM DPI

The Current Medium Document Production Instruction shall be represented by a CURRENT MEDIUM DPI structure element. A CURRENT MEDIUM DPI structure element shall be a base structure element. The value of a CURRENT MEDIUM DPI structure element shall be of type *Name*.

12.7.2 Current Medium Document Production Instruction Processing

The effect of the Current Medium Document Production Instruction shall be to set the value of the *current-medium* presentation parameter to the value of the CURRENT MEDIUM DPI for the scope of the Current Medium Document Production Instruction.

12.7.3 Current Medium Document Production Instruction Reconciliation

The Current Medium Document Production Instruction is meaningful in any PAGE or PAGESET or in Supplementary DPI. If a Current Medium Document Production Instruction occurs in both a PAGESET and in a subor-

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642582 975

dinate PAGE or PAGESET, the Current Medium Document Production Instruction in the subordinate PAGE or PAGESET shall take precedence. If a Current Medium Document Production Instructions occurs in both Supplementary DPI and Document DPI, the Current Medium Document Production Instructions in the Document DPI shall take precedence for their scope.

12.7.4 Current Medium Document Production Instruction Fallback

There is no fallback required for the Current Medium Document Production Instruction.

13 Presentation Document Production Instructions

Presentation Document Production Instructions affect the process of rendering the page images and their placement on the media. This includes the selection of the page images to be presented, the placement of individual page images on instances of the medium, the number of copies to be printed, and the delivery of the document to the user.

The presentation parameters which are affected by the Presentation Document Production Instructions are shown in table 19.

Table 19

Name	Affects
number-of-copies	number of copies to be printed
page-select	selects PAGEs of a PAGESET included in the document
supplementary-page-select	selects PAGEs of a document to be printed
current-side	placement of page image on obverse or reverse side
sides	number of sides of two sided sheet media to be used
plex	placement of page images on instances of the media
x-image-shift	initial *CurrentTransformation*
y-image-shift	initial *CurrentTransformation*

13.1 Copies Document Production Instruction

The Copies Document Production Instruction specifies the number of copies of the document to be printed.

13.1.1 COPIES DPI

The Copies Document Production Instruction shall be represented by a COPIES DPI structure element. A COPIES DPI structure element shall be a base structure element. The value of a COPIES DPI structure element shall be of type *Cardinal*.

13.1.2 Copies Document Production Instruction Processing

The effect of the Copies Document Production Instruction shall be to set the value of the *number-of-copies* presentation parameter to the value of the COPIES DPI structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642583 801

Copies of the document shall be presented to the user in a manner consistent with the values of the *output-bin* and *output-specification* presentation parameters. If the value of the *number-of-copies* presentation parameter is zero, the document shall be processed but no copies shall be printed. The default value of the *number-of-copies* presentation parameter shall be one.

13.1.3 Copies Document Production Instruction Reconciliation

The Copies Document Production Instruction is meaningful only in a DOCUMENT or in Supplementary DPI. Any Copies Document Production Instruction which occurs in a BLOCK which is not a DOCUMENT shall be ignored. If a Copies Document Production Instruction occurs in both Document DPI and Supplementary DPI, the Supplementary DPI shall take precedence.

13.1.4 Copies Document Production Instruction Fallback

A system which cannot present multiple copies of the document in collated form shall indicate failure to do so by generating a structure warning. If the value of the *abort-policy* presentation parameter allows for continued processing, the fallback behavior shall be to present the pages in either

— uncollated form (all copies of the first page followed by all copies of the second page, etc.);

— in block collated form (all copies of the first *n* pages (in collated form), followed by all copies of the next *n* pages, etc.) for some system dependent value of *n*.

The choice of uncollated or block collated form and the block size shall be system dependent.

13.2 Page Select Document Production Instruction

The Page Select Document Production Instruction specifies which PAGEs of a PAGESET are to be included as part of the document. The default behavior (if there is no Page Select Document Production Instruction in a PAGESET), shall be that all the PAGEs in the PAGESET shall be included.

13.2.1 PAGE SELECT DPI

The Page Select Document Production Instruction shall be represented by a PAGE SELECT DPI structure element. A PAGE SELECT DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

— one or more PAGE SELECTION structure elements.

13.2.1.1 PAGE SELECTION

A PAGE SELECTION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one START PAGE structure element;

— one END PAGE structure element.

These structure elements may occur in any order.

For each PAGE SELECTION structure element, the value of the immediately subordinate END PAGE structure element shall be greater than or equal to the value of START PAGE structure element, and the value of the immediately subordinate START PAGE structure element shall be larger than the value of the END PAGE structure element which is subordinate to the immediately preceding PAGE SELECTION (if any).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

13.2.1.2 START PAGE

A START PAGE structure element shall be a base structure element. The value of a START PAGE structure element shall be of type *Positive Integer*.

13.2.1.3 END PAGE

An END PAGE structure element shall be a base structure element. The value of an END PAGE structure element shall be of type *Positive Integer*.

13.2.2 Page Select Document Production Instruction Processing

The effect of a Page Select Document Production Instruction shall be to set the value of the *page-select* presentation parameter for the most immediately superior PAGESET.

If the value of the the START PAGE which is subordinate to the nth PAGE SELECTION is *start-page*$_n$, and the value of the the END PAGE which is subordinate to the nth PAGE SELECTION is *end-page*$_n$, then the value of *page-select* shall be obtained by:

a) setting the value of the *page-select* presentation parameter to

$$page\text{-}select(i) = false$$

for all values of i;

b) for each PAGE SELECTION, setting

$$page\text{-}select(i) = true$$

for all values of i for which *start-page*$_n$ σισ *end-page*$_n$.

If there are no subordinate PAGE SELECTION structure elements, the value of the *page-select* presentation parameter shall be $page\text{-}select(i) = false$ for all values of i.

If the processing of the PAGESET (including the processing of Page Select Document Production Instructions of any subordinate PAGESETs) generates n PAGEs, then for 1σισn the ith PAGE in the sequential order of the PAGESET

— included if $page\text{-}select(i) = true$;

— shall be excluded by the Page Select Document Production Instruction if $page\text{-}select(i) = false$.

The sequence of PAGEs which are included (i.e. for which $page\text{-}select(i) = true$) shall constitute the sequence of PAGEs of the PAGESET The default value of *page-select* for each PAGESET shall be $page\text{-}select(i) = true$ for all values of i.

There is no fallback behavior for page selection

13.2.3 Page Select Document Production Instruction Reconciliation

The Page Select Document Production Instruction is meaningful only in a PAGESET. A PAGE SELECT DPI which occurs in a PICTURE or in Supplementary DPI shall be ignored.

If a Page Select Document Production Instruction occurs in both a PAGESET and in a subordinate PAGE or PAGESET, the set of PAGEs which are part of the subordinate PAGESET shall be those PAGEs for which the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

value of *page-select(i)* for the subordinate PAGE SET is *true*. The effect of the Page Select Document Production Instruction of the superior PAGESET shall be to make a further selection from the pages for which the value of *page-select(i)* for the subordinate PAGESET is *true*.

13.2.4 Page Select Document Production Instruction Fallback

There is no fallback required for the Page Select Document Production Instruction.

13.3 Supplementary Page Select Document Production Instruction

The Supplementary Page Select Document Production Instruction specifies which pages of a document are to be presented in an instance of presentation. The Supplementary Page Select Document Production Instruction acts on the set of pages which are selected by Page Select Document Production Instructions in the Document.

13.3.1 SUPPLEMENTARY PAGE SELECT DPI

The Supplementary Page Select Document Production Instruction shall be represented by a SUPPLEMENTARY PAGE SELECT DPI structure element. A SUPPLEMENTARY PAGE SELECT DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

— one or more PAGE SELECTION structure elements.

13.3.2 Supplementary Page Select Document Production Instruction Processing

The effect of a Supplementary Page Select Document Production Instruction shall be to set the value of the *supplementary-page-select* presentation parameter.

If the value of the the START PAGE which is subordinate to the nth PAGE SELECTION is $start\text{-}page_n$, and the value of the the END PAGE which is subordinate to the nth PAGE SELECTION is $end\text{-}page_n$, then the value of *page-select* shall be obtained by:

a) setting the value of the *supplementary-page-select* presentation parameter to

$$supplementary\text{-}page\text{-}select(i) = false$$

for all values of i;

b) for each PAGE SELECTION, setting

$$supplementary\text{-}page\text{-}select(i) = true$$

for all values of i for which $start\text{-}page_n \sigma i \sigma end\text{-}page_n$.

If there are no subordinate PAGE SELECTION structure elements, the value of the *page-select* presentation parameter shall be $page\text{-}select(i) = false$ for all values of i.

If the processing of the DOCUMENT generates n PAGEs, then for $1 \sigma i \sigma n$ the ith PAGE

— shall be presented if $supplementary\text{-}page\text{-}select(i) = true$;

— shall not be presented if $supplementary\text{-}page\text{-}select(i) = false$.

The default value of *supplementary-page-select* shall be $supplementary\text{-}page\text{-}select(i) = true$ for all values of i.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642586 510

13.3.3 Supplementary Page Select Document Production Instruction Reconciliation

The Supplementary Page Select Document Production Instruction is meaningful only in Supplementary DPI. Any Supplementary Page Select Document Production Instruction which occurs in Document DPI shall be ignored.

13.3.4 Supplementary Page Select Document Production Instruction Fallback

There is no fallback required for the Supplementary Page Select Document Production Instruction.

13.4 Sides Document Production Instruction

The Sides Document Production Instruction specifies whether page images are to be printed on one or two sides of a sheet medium (such as paper). It is meaningful only for the presentation of documents on such media.

13.4.1 SIDES DPI

The Sides Document Production Instruction is represented by a SIDES DPI structure element. A SIDES DPI structure element shall be a base structure element. The value of a SIDES DPI structure element shall be of type *Cardinal*, and shall be either 1 or 2.

13.4.2 Sides Document Production Instruction Processing

The effect of the Sides Document Production Instruction shall be to set the value of the *sides* presentation parameter to the value of the SIDES DPI. The default value of the *sides* presentation parameter shall be 1. The effect of the *sides* presentation parameter on the placement of page images on instances of the media is specified in 16.6 .

13.4.3 Sides Document Production Instruction Reconciliation

The Sides Document Production Instruction is meaningful in any PAGESET or in Supplementary DPI. If a Sides Document Production Instruction occurs in both a PAGESET and in a subordinate PAGESET, the Sides Document Production Instruction in the subordinate PAGESET shall take precedence for its scope. If a Sides Document Production Instruction occurs in both Document DPI and Supplementary DPI, the Sides Document Production Instruction in Supplementary DPI shall take precedence.

13.4.4 Sides Document Production Instruction Fallback

A system which does not support imaging on two sides of the selected sheet medium shall indicate failure to do so by generating a structure warning if the value of the SIDES DPI is 2. If the value of the *abort-policy* presentation parameter allows for continued processing, the fallback behavior shall be to present each page image on a separate instance of the selected sheet medium.

13.5 Plex Document Production Instruction

The Plex Document Production Instruction specifies the relationship between page images and instances of the media. It is intended for the presentation of documents on two sides of sheet media such as paper or for the placement of single page images on instances of the media so that they can be copied onto two sides of sheet media.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642587 457

13.5.1 PLEX DPI

The Plex Document Production Instruction is represented by a PLEX DPI structure element. A PLEX DPI structure element shall be a base structure element. The value of a PLEX DPI structure element shall be of type *Public Object Identifier*. Public Object Identifier values for PLEX DPI are defined by ISO/IEC 10175 as standard values for the *plex* attribute. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

13.5.2 Plex Document Production Instruction Processing

The effect of the Plex Document Production Instruction shall be to set the value of the *plex* presentation parameter to the value of the PLEX DPI structure element. The default value of the *plex* presentation parameter shall be *simplex*. The effect of the *plex* presentation parameter on the presentation of SPDL documents is specified in 16 6 and 16.7 .

13.5.3 Plex Document Production Instruction Reconciliation

The Plex Document Production Instruction is meaningful in any PAGESET or in Supplementary DPI. If a Plex Document Production Instruction occurs in both a PAGESET and in a subordinate PAGESET, the Plex Document Production Instruction in the subordinate PAGESET shall take precedence for its scope. If a Plex Document Production Instruction occurs in both Document DPI and Supplementary DPI, the Plex Document Production Instruction in Supplementary DPI shall take precedence.

13.5.4 Plex Document Production Instruction Fallback

Fallback behavior for printers which can print only on one side of the medium is specified in 16.6.

13.6 X-Image-Shift Document Production Instruction

The X-Image-Shift Document Production Instruction specifies an initial value for the *CurrentTransformation* which shifts the page image toward or away from the edge of the physical medium along which the *y*-axis of the Reference Coordinate System is placed. It is intended to be used to offset the page image to allow for a later binding operation.

13.6.1 X-IMAGE SHIFT DPI

The X-Image-Shift Document Production Instruction is represented by an X-IMAGE SHIFT DPI structure element. An X-IMAGE SHIFT DPI structure element shall be a base structure element. The value of an X-IMAGE SHIFT DPI structure element shall be of type *Number*.

13.6.2 X-Image-Shift Document Production Instruction Processing

The effect of the X-Image-Shift Document Production Instruction shall be to set the value of the *x-image-shift* presentation parameter to the value of the X-IMAGE SHIFT DPI structure element. The effect of the *x-image-shift* presentation parameter on the positioning of the Reference Coordinate System on the medium is specified in 16.7. The default value of *x-image-shift* shall be zero, which results in no net change to the *CurrentTransformation*.

13.6.3 X-Image-Shift Document Production Instruction Reconciliation

The X-Image-Shift Document Production Instruction is meaningful in any PAGE or PAGESET, or in Supplementary DPI. If X-Image-Shift Document Production Instructions occur in both a PAGESET and in a subordinate PAGE or PAGESET, the X-Image-Shift Document Production Instruction in the subordinate PAGE or PAGESET shall take precedence for its scope If a X-Image-Shift Document Production Instruction occurs in both Docu-

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642588 393

ment DPI and Supplementary DPI, the X-Image-Shift Document Production Instruction in Supplementary DPI shall take precedence.

13.6.4 X-Image-Shift Document Production Instruction Fallback

There is no fallback required for the X-Image-Shift Document Production Instruction.

13.7 Y-Image-Shift Document Production Instruction

The Y-Image-Shift Document Production Instruction specifies an initial value for the *CurrentTransformation* which shifts the page image toward or away from the edge of the physical medium along which the *y*-axis of the Reference Coordinate System is placed. It is intended to be used to offset the page image to allow for a later binding operation.

13.7.1 Y-IMAGE SHIFT DPI

The Y-Image-Shift Document Production Instruction is represented by a Y-IMAGE SHIFT DPI structure element. A Y-IMAGE SHIFT structure element shall be a base structure element. The value of the Y-IMAGE SHIFT DPI structure element shall be of type *Number*

13.7.2 Y-Image-Shift Document Production Instruction Processing

The effect of the Y-Image-Shift Document Production Instruction shall be to set the value of the *y-image-shift* presentation parameter to the value of the Y-IMAGE SHIFT DPI structure element. The effect of the *y-image-shift* presentation parameter on the positioning of the Reference Coordinate System on the medium is specified in 16.7. The default value of *y-image-shift* shall be zero, which results in no net change to the *CurrentTransformation*.

13.7.3 Y-Image-Shift Document Production Instruction Reconciliation

The Y-Image-Shift Document Production Instruction is meaningful in any PAGE or PAGESET, or in Supplementary DPI. If a Y-Image-Shift Document Production Instruction occurs in both a PAGESET and in a subordinate PAGE or PAGESET, the Y-Image-Shift Document Production Instruction in the subordinate PAGE or PAGESET shall take precedence for its scope. If a Y-Image-Shift Document Production Instruction occurs in both Document DPI and Supplementary DPI, the Y-Image-Shift Document Production Instruction in Supplementary DPI shall take precedence.

13.7.4 Y-Image-Shift Document Production Instruction Fallback

There is no fallback required for the Y-Image-Shift Document Production Instruction.

13.8 Current Side Document Production Instruction

The Current Side Document Production Instruction specifies the side of a two sided medium on which a particular page is to be printed

13.8.1 CURRENT SIDE DPI

The Current Side Document Production Instruction is specified by a CURRENT SIDE DPI structure element. A CURRENT SIDE DPI structure element shall be a base structure element. The value of a CURRENT SIDE DPI structure element shall be of type *Cardinal*, and shall be either 1 or 2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

13.8.2 Current Side Document Production Instruction Processing

The effect of a Current Side Document Production Instruction shall be to set the value of the *current-side* presentation parameter associated with the PAGE in which the CURRENT SIDE DPI occurs to the value of the CURRENT SIDE DPI. The effect of the *current-side* presentation parameter is specified in 16.6 and 16.7.

13.8.3 Current Side Document Production Instruction Reconciliation

The Current Side Document Production Instruction is meaningful in any PAGE or PAGESET. The effect of a Current Side Document Production in a PAGESET shall be the same as the occurrence of a Current Side Document Production Instruction in the first PAGE of that PAGESET. Any Current Side Document Production Instruction which occurs in a PICTURE which is not a PAGE or as a Supplementary DPI shall be ignored.

13.8.4 Current Side Document Production Instruction Fallback

There is no fallback required for the Current Side Document Production Instruction.

14 Finishing Document Production Instruction

The Finishing Document Production Instruction specifies an interface to a comprehensive and extensible set of finishing processes. The following presentation parameters are affected by the Finishing Document Production Instruction as shown in table 20.

Table 20

Name	Affects
finishing-processes	defines a list of Finishing Processes to be applied
reference-size	affects Finishing Processes based on document size
reference-edge	sheet alignment and location and orientation of Process Axis
jog-edge	sheet alignment
process-offset	distance of Process Axis from Reference Edge
head-locations	location of heads on Process Axis

14.1 Finishing Model

The term *finishing* refers to the final document production processes applied to a document subsequent to its imaging and prior to its delivery. Finishing processes are normally applied to documents which are presented on one or more instances of rectangular sheet media such as paper. The following definitions refer to the instances of media on which the document has been presented as "sheets", and assume that all of the presentation surfaces used for document presentation are on rectangular sheet media. This International Standard defines the application of finishing processes to such documents only. The application of finishing processes to documents which have been presented on other media is system dependent.

Finishing is applied by a collection of processes which are commonly provided by a traditional print shop bindery. Each of these processes is referred to as a Finishing Process. This International Standard provides a Finishing Document Production Instruction identifies Finishing Processes to be applied

A *Finishing Process* is typically applied to a document by a machine. Examples of Finishing Processes include process such as trimming a document, folding the sheets in a document, and applying binding to a document.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Conceptually, Finishing Processes are applied to document one at a time in a specified sequence. In this International Standard, each Finishing Process and all of the parameters required to specify its application are collected into a structure element which is referred to as a Finishing Specification.

Multiple Finishing Processes may be specified by multiple Finishing Specifications. Finishing Specifications are serialized in the value of the Finishing Document Production Instruction according to their intended sequence of application. This sequencing is significant. For example, the effect of first folding the sheets in a document and then trimming will yield a different result than trimming the sheets and then folding them.

A Finishing Process is applied to a document by one or more physical mechanisms. For example, a Finishing Process which applies a corner staple may use only one physical mechanism for stapling, whereas a Finishing Process which applies a three-hole punch may use three physical mechanisms. Each physical mechanism used in the application of Finishing Process is referred to as a head.

In some machines which apply a Finishing Process, head positioning is fixed, whereas with more capable machines head positions can be moved subject only to certain machine-specific physical constraints. However, even in the case of more capable machines, heads can usually be positioned only in a line relative to each other. This physical constraint on head positioning means that heads cannot be positioned arbitrarily. The specification of the Finishing Document Production Instruction in this International Standard constrains the ability to specify head positions in a manner which is consistent with most physical systems. The finishing process presentation parameters are shown in figure 6.

Heads are positioned along a Process Axis, which is an imaginary on-center line which transects each head. The exact location at which the line transects a given head is formally unspecified since it is specific to the Finishing Process. However, the position at which the Process Axis transects each of the heads associated with the Finishing Process is consistent across each of the head mechanisms.

The Process Axis is used as a reference for positioning heads associated with a Finishing Process. This makes explicit the constraint that for any single Finishing Process head positioning must be along a single Process Axis.

While head motion perpendicular to the Process Axis is constrained, linear motion along the Process Axis is not formally constrained. The only constraint on linear motion of heads along the Process Axis is subject to the physical capabilities of the machine.

The edge of the document relative to which the Finishing Process is applied is referred to as the Reference Edge. Since bindery systems can typically reposition a document to any edge, it would not be sufficient to constrain the Reference Edge to a static edge of the document The Reference Edge can be any edge of the document. Documents are therefore repositioned implicitly through changing the Reference Edge associated with a given Finishing Process. This International Standard provides an explicit presentation parameter of the Finishing Document Production Instruction, the *reference-edge* presentation parameter, in order to enable this flexible positioning.

When a document consists of multiple sheets of different sizes, it is necessary to specify the relative locations of the sheets for finishing. This is accomplished by one convention and one additional parameter.

The convention is that for Finishing Processes all of the sheets are aligned so that the edges of the sheets which correspond to the Reference Edge coincide. For example, if the Reference Edge is the left edge, all of the sheets will be aligned so that the *y*-axis of the medium Reference Coordinate Systems (of the obverse side of the sheet if printing on two sides of the medium) coincides.

The additional parameter is the *jog-edge* presentation parameter. The value of the *jog-edge* presentation parameter is one of the two edges which is perpendicular to the Reference Edge. When the *jog-edge* presentation parameter is specified, all of the sheets are aligned so that the edges of the sheets which correspond to the the value of the *jog-edge* presentation parameter coincide.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

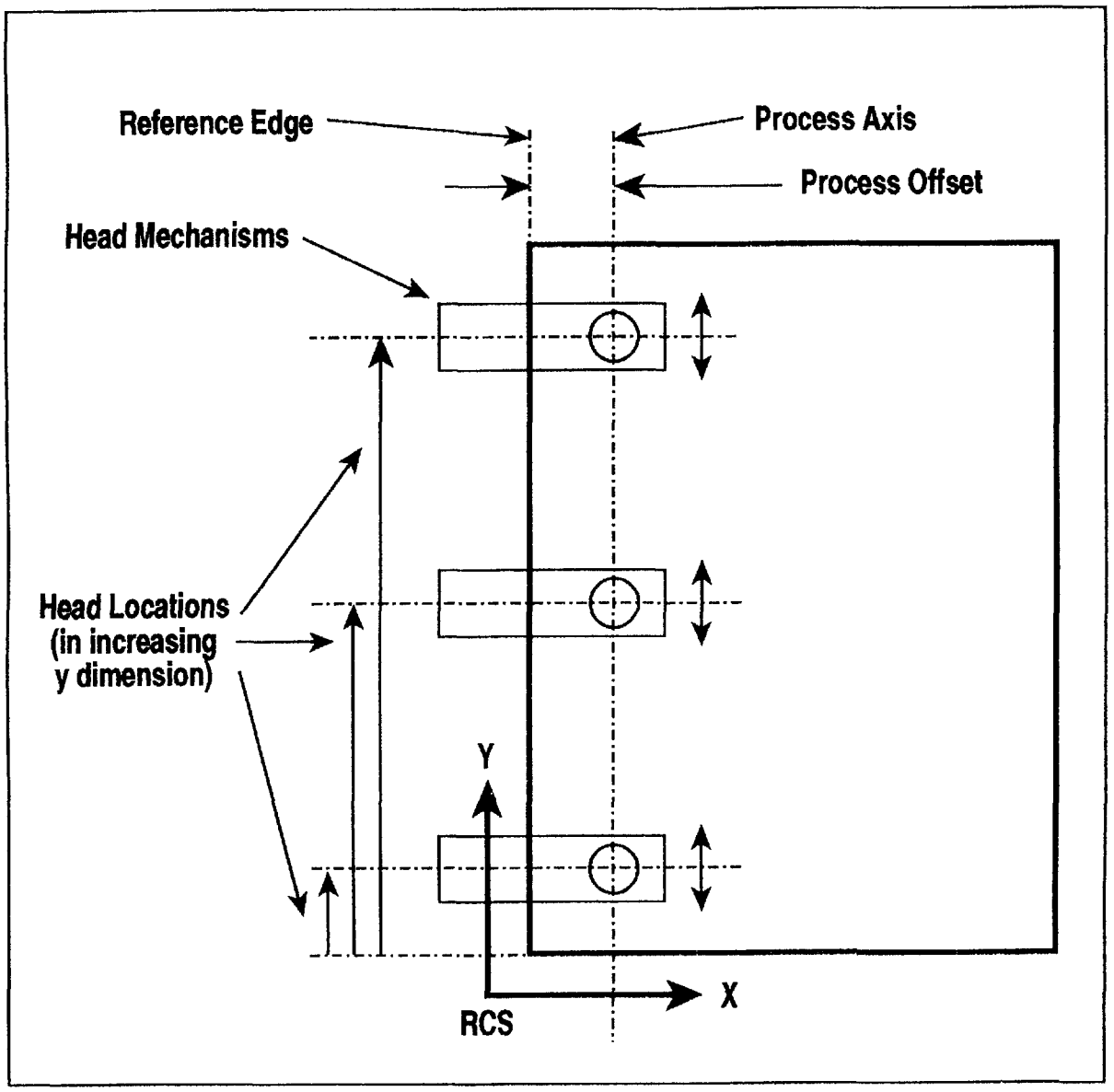

Figure 6 — Finishing Process Presentation Parameters

The parameters required for a Finishing Process may be specified by either an explicit set of values or by a Public Object Identifier which identifies a set of parameters. There are no Public Object Identifier values defined by this International Standard which identify sets of Finishing Process presentation parameters.

14.2 Finishing Process Presentation Parameters

Each Finishing Process has associated with it several presentation parameters. The values of these presentation parameters may be specified by the FINISHING SPECIFICATION structure element which specifies the Finishing Process, inherited from previous Finishing Processes, or default values specific to the Finishing Process.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642592 814

14.2.1 Inherited Finishing Process Presentation Parameters

There are three presentation parameters whose values may be inherited from one Finishing Process to another. These are:

— *reference-size*

— *reference-edge*

— *jog-edge*

The values of these presentation parameters which apply to a particular Finishing Process are determined as follows:

a) At the beginning of the Finishing Document Production Instruction, the presentation parameters which may be inherited are initialized to their default values. The set of default presentation parameters is the set of presentation parameters prior to the processing of the first FINISHING PROCESS SPECIFICATION.

b) Each FINISHING PROCESS SPECIFICATION may specify a value for one or more of these presentation parameters. The set of presentation parameters which apply to a particular Finishing Process is the set of presentation parameters prior to the processing of the FINISHING PROCESS SPECIFICATION which specifies that Finishing Process modified by any presentation parameter specifications in the FINISHING PROCESS SPECIFICATION.

c) The execution of the Finishing Process itself may change the value of one or more of the presentation parameters. The set of presentation parameters at the end of the Finishing Process is the set of presentation parameters which was applicable to the Finishing Process modified by any changes which result from the Finishing Process itself.

d) The set of presentation parameters at the end of a Finishing Process is the set of presentation parameters prior to the processing of the next FINISHING PROCESS SPECIFICATION (if any).

14.2.2 Other Finishing Process Parameters

Two other presentation parameters are set independently for each Finishing Process. These process parameters are:

— *process-offset*

— *head-locations*

There are no default values for these presentation parameters.

14.2.3 Reference Size

The value of the *reference-size* presentation parameter specifies the nominal medium size in millimeters for all calculations in the associated Finishing Process. The meaning of a value of (x, y) is that the Finishing Process shall be as defined for a document consisting of sheets whose size is x mm by y mm, for which the medium Reference Coordinate System is associated with a sheet in such a way that:

— one of the edges of length x mm corresponds to the positive x-axis of the medium Reference Coordinate System;

— one of the edges of length y mm corresponds to the positive y-axis of the medium Reference Coordinate system.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The value of the *reference-size* presentation parameter is specified by a REFERENCE SIZE structure element. A REFERENCE SIZE structure element shall be an XYDIMENSIONS structure element. The default value of the *reference-size* presentation parameter shall be the Medium Size of the first medium selected by the Page Select Document Production Instruction.

14.2.4 Reference Edge

The *reference-edge* presentation parameter specifies the edge of the document relative to which a Finishing Process is applied. The *reference-edge* presentation parameter also specifies the alignment of the sheets on which the document is presented: all sheets shall be aligned so that the edges of the sheets which correspond to the Reference Edge coincide. The resulting edge shall be referred to as the Reference Edge of the document.

The *reference-edge* presentation parameter is specified by a REFERENCE EDGE structure element. A REFERENCE EDGE structure element shall be an EDGE structure element. The default value of the *reference-edge* presentation parameter shall be *left edge*.

14.2.5 Jog Edge

The value of the *jog-edge* presentation parameter specifies a second edge of the document along which instances of the media are aligned. The value of the *jog-edge* presentation parameter shall be one of the two edges which are perpendicular to the Reference Edge. The edge which is identified by the value of the *jog-edge* presentation parameter is called the Jog Edge. All sheets shall be aligned so that the edges of the sheets which correspond to the Jog Edge coincide.

The the value of the *jog-edge* presentation parameter is specified by a JOG EDGE structure element. A JOG EDGE structure element shall be an EDGE structure element. The default value of the *jog-edge* presentation parameter depends on the value of the *reference-edge* presentation parameter:

— if the value of *reference-edge* is *left edge* or *right edge*, the default value of *jog-edge* shall be *bottom edge*;

— if the value of *reference-edge* is *top edge* or *bottom edge*, the default value of *jog-edge* shall be *left edge*.

14.2.6 Process Offset

The *process-offset* presentation parameter specifies the offset from the Reference Edge to the Process Axis in millimeters. Changes to the *process-offset* presentation parameter value cause all of the heads along the Process Axis to be repositioned uniformly. The value of the *process-offset* presentation parameter shall be a Non-Negative Number.

The *process-offset* presentation parameter is specified by a PROCESS OFFSET structure element. A PROCESS OFFSET structure element shall be a DIMENSION structure element.

14.2.7 Head Locations

The *head-locations* presentation parameter specifies a set of locations at which one or more heads apply a Finishing Process. Heads are on-center relative to the Process Axis, and are positioned linearly along the Process Axis as specified in the value of the *head-locations* finishing property.

The value of the *head-locations* presentation parameter shall be a sequence of Non-Negative Numbers. If the Reference Edge is either the top edge or the bottom edge, then the sequence of values $x_1, \ldots, x_n$ shall represent offsets from the *x*-axis of the medium Reference Coordinate System. If the Reference Edge is either the left edge or the right edge, then the sequence of values $x_1, \ldots, x_n$ shall represent offsets from the *y*-axis of the medium Reference Coordinate System.

The *head-locations* presentation parameter is specified by a HEAD LOCATIONS structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642594 697

14.3 Finishing Document Production Instruction

Finishing Document Production Instructions may specify the set of Finishing Processes as either:

— a named set of Finishing Processes, specified by an Environment Identifier;

— a list of Finishing Processes specified by a FINISHING PROCESS LIST.

Individual Finishing Processes are specified by FINISHING PROCESS SPECIFICATION structure elements. Each FINISHING PROCESS SPECIFICATION specifies a Finishing Process together with presentation parameters which apply to that Finishing Process.

Many FINISHING PROCESS SPECIFICATION structure elements allow the specification a Finishing Process as either·

— a named Finishing Process specified by an Environment Identifier;

— a parameterized Finishing Process specified by a composite structure element.

The parameters of a Parameterized Finishing Process are specified by a composite structure element. The immediate subordinates of the composite structure element depend on the Finishing Process. All of the parameters of a named Finishing Process are specified by a the Environment Identifier which identifies the Finishing Process.

14.3.1 FINISHING DPI

Finishing Document Production Instructions are represented by a FINISHING DPI structure element. A FINISHING DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one FINISHING MESSAGE structure element;

— one FINISHING SPECIFICATION structure element.

The immediate subordinates shall occur in the above order.

14.3.1.1 FINISHING MESSAGE

A FINISHING MESSAGE structure element shall be a base structure element. The value of a FINISHING MESSAGE structure element shall be of type *Printable String*.

A FINISHING MESSAGE shall have no effect on document presentation, but may effect printing management. Any effect of a FINISHING MESSAGE on printing management shall be system dependent. FINISHING MESSAGE structure elements may be ignored by the Presentation Process.

14.3.1.2 FINISHING SPECIFICATION

A FINISHING SPECIFICATION structure element shall be one of:

— a NAMED FINISHING structure element,

— a FINISHING SPECIFICATION LIST structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

14.3.1.3 NAMED FINISHING

A NAMED FINISHING structure element shall be a base structure element. The value of a NAMED FINISHING structure element shall be of type *Environment Identifier*.

The value of a NAMED FINISHING structure element may identify a single Finishing Process or a sequence of Finishing Processes. If the value of a NAMED FINISHING structure element identifies a single Finishing Process, such as a named stitching Finishing Process, the meaning of that value shall be that the sequence of Finishing Processes specified consists of the single Finishing Process identified by the value of the NAMED FINISHING structure element .

Public Object Identifier values defined by ISO/IEC 10175 which identify single named Finishing Processes, and the meaning of those values, are described in annex F (informative). There are no additional Public Object Identifier values for NAMED FINISHING which are defined by this International Standard.

14.3.1.4 FINISHING SPECIFICATION LIST

A FINISHING SPECIFICATION LIST structure element shall be a composite structure element which shall have the following immediate subordinates:

— *one or more* FINISHING PROCESS SPECIFICATION structure elements.

14.3.2 Finishing Document Production Instruction Processing

The effect of the Finishing Document Production Instruction is to set the value of the *finishing-processes* parameter to the sequence of Finishing Processes specified by the sequence of FINISHING SPECIFICATIONs. The order of these Finishing Processes shall be the same as the order of the corresponding FINISHING SPECIFICATIONs.

The *finishing-processes* parameter specifies the sequence of Finishing Processes that shall be applied during document finishing. The *finishing-processes* parameter may specify a sequence of Finishing Processes. In this case, each Finishing Process shall be performed in the sequence in which it is occurs in the value of the *finishing-processes* parameter. The default value of the *finishing-processes* parameter is a null list, which specifies that no Finishing Processes are to be applied.

14.3.3 Finishing Document Production Instruction Reconciliation

The Finishing Document Production Instruction is meaningful only in a DOCUMENT or in Supplementary DPI. If the Finishing Document Production Instruction occurs as both Document DPI and Supplementary DPI, the Supplementary DPI shall take precedence.

14.3.4 Finishing Document Production Instruction Fallback

A system which cannot perform one or more of the Finishing Processes specified by FINISHING SPECIFICATIONs shall indicate failure to do so by generating a structure warning. If the value of the *abort-policy* parameter allows for continued processing after generation of the structure warning, the effect of the structure warning shall be system dependent.

14.4 FINISHING PROCESS SPECIFICATION

Finishing Processes are specified by FINISHING PROCESS SPECIFICATION structure elements. Each FINISHING PROCESS SPECIFICATION specifies a Finishing Process together with presentation parameters which apply to that Finishing Process.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642596 46T

Many FINISHING PROCESS SPECIFICATION structure elements allow the specification a Finishing Process as either:

— a named Finishing Process specified by an Environment Identifier;

— a parameterized Finishing Process specified by a composite structure element.

The parameters of a Parameterized Finishing Process are specified by a composite structure element. The immediate subordinates of the composite structure element depend on the Finishing Process. All of the parameters of a named Finishing Process are specified by a the *Environment Identifier* which identifies the Finishing Process.

A FINISHING PROCESS SPECIFICATION structure element shall be one of:

— a STITCHING SPECIFICATION structure element;

— a BINDING SPECIFICATION structure element;

— a TRIMMING SPECIFICATION structure element;

— a DIE CUTTING SPECIFICATION structure element;

— a PUNCHING SPECIFICATION structure element;

— a PERFORATING SPECIFICATION structure element;

— a SLITTING SPECIFICATION structure element;

— an INSERT SPECIFICATION structure element;

— a COVER SPECIFICATION structure element;

— a FOLDING SPECIFICATION structure element;

— an OTHER FINISHING SPECIFICATION structure element.

14.5 STITCHING SPECIFICATION

A STITCHING SPECIFICATION shall be either:

— a NAMED STITCHING structure element;

— a STITCHING PARAMETERS structure element.

14.5.1 NAMED STITCHING

A NAMED STITCHING structure element shall be a base structure element. The value of a NAMED STITCHING structure element shall be of type *Environment Identifier*.

The value of a NAMED STITCHING structure element shall identify a stitching operation. Public Object Identifier values for NAMED STITCHING are defined by ISO/IEC 10175 as standard values for the attribute *stitching*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642597 3T6

14.5.2 STITCHING PARAMETERS

A STITCHING PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— zero or one PROCESS OFFSET structure element;

— zero or one HEAD LOCATIONS structure element;

— zero or one STITCH TYPE structure element.

The immediate subordinates may occur in any order.

14.5.3 STITCH TYPE

A STITCH TYPE structure element shall be a base structure element. The value of a STITCH TYPE structure element shall be of type *Environment Identifier*.

The value of a STITCH TYPE structure element shall identify a type of stitching or fastening. These values include stapling as well as stitching with thread. Public Object Identifier values for STITCH TYPE are defined by ISO/IEC 10175 as standard values for the attribute *stitch-type*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

The default value of STITCH TYPE shall be *dpi::stitching::type::default*.

14.5.4 Stitching Finishing Process

A STITCHING SPECIFICATION specifies a Finishing Process which applies one or more stitches to the document. It is a general specification for invoking application of various types of stitches, including staples and brads as well as sewn stitches. Specially positioned types of stitching such as saddle stitching can also be specified. The type of stitching can be separated into two general types:

— discrete stitching, which consists of placing one or more discrete stitches, staples, or similar fastenings at one or more head locations along the Process axis;

— continuous stitching, which consists of placing a continuous line of stitching along the Process Axis.

If the STITCHING SPECIFICATION specifies a *process-offset* which specifies head positioning outside of the achievable range, the system shall use a default offset and a structure warning shall be generated.

14.6 BINDING SPECIFICATION

The binding operation specifies a Finishing Process which binds the document along the reference edge. The binding material is specified by the combination of the Binding Type and Binding Color finishing properties.

A BINDING SPECIFICATION shall be either

— a NAMED BINDING structure element;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— a BINDING PARAMETERS structure element.

14.6.1 NAMED BINDING

A NAMED BINDING structure element shall be a base structure element. The value of a NAMED BINDING structure element shall be of type *Environment Identifier*.

The value of a NAMED BINDING structure element shall identify a binding operation. Public Object Identifier values for NAMED BINDING are defined by ISO/IEC 10175 as standard values for the attribute *binding*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

14.6.2 BINDING PARAMETERS

A BINDING PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— zero or one BINDING TYPE structure element;

— zero or one BINDING COLOR structure element.

The immediate subordinates may occur in any order.

14.6.3 BINDING TYPE

A BINDING TYPE structure element shall be a base structure element. The value of a BINDING TYPE structure element shall be of type *Environment Identifier*.

The value of a BINDING TYPE structure element shall identify a type of binding to be applied to the document. Public Object Identifier values for BINDING TYPE are defined by ISO/IEC 10175 as standard values for the attribute *binding-type*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

The default value of BINDING TYPE shall be *dpi::binding::type::default*.

14.6.4 BINDING COLOR

A BINDING COLOR structure element shall be a base structure element. The value of a BINDING COLOR structure element shall be of type *Environment Identifier*.

The value of a BINDING COLOR structure element shall identify the color of the binding to be applied. Public Object Identifier values for BINDING COLOR are defined by ISO/IEC 10175 as standard values for the attribute *binding-colour*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

The default value of BINDING COLOR shall be *dpi::binding::color::default*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

14.6.5 Binding Finishing Process

A BINDING SPECIFICATION specifies a Finishing Process which binds the document along the reference edge. The binding material is specified by the combination of the Binding Type and Binding Color finishing properties.

14.7 TRIMMING SPECIFICATION

A TRIMMING SPECIFICATION shall be either:

— a NAMED TRIMMING structure element;

— a TRIMMING PARAMETERS structure element.

14.7.1 NAMED TRIMMING

A NAMED TRIMMING structure element shall be a base structure element. The value of a NAMED TRIMMING structure element shall be of type *Environment Identifier*.

The value of a NAMED TRIMMING structure element shall identify a method of trimming the document. Public Object Identifier values for NAMED TRIMMING are defined by ISO/IEC 10175 as standard values for the attribute *trimming*. The Object Names for which Public Object Identifier values are defined by ISO/IEC 10175 are provided in annex F (informative).

14.7.2 TRIMMING PARAMETERS

A TRIMMING PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— one TRIM OFFSET structure element;

— one TRIM DIMENSIONS structure element.

The immediate subordinates may occur in any order.

14.7.3 TRIM OFFSET

The TRIM OFFSET structure element shall be a DIMENSION structure element. The default value for TRIM OFFSET shall be zero.

14.7.4 TRIM DIMENSIONS

The TRIM DIMENSIONS structure element shall be an XYDIMENSIONS structure element. The default value for TRIM DIMENSIONS shall be the value of the *reference-size* presentation parameter (after processing any REFERENCE SIZE structure element which is immediately subordinate to the TRIMMING SPECIFICATION).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642600 710

14.7.5 Trimming Finishing Process

A TRIMMING SPECIFICATION specifies a Finishing Process which cuts all the edges of the set of sheets being finished except the edge specified by the value of *reference-edge*. The effect of trimming shall always be to trim along three edges regardless of whether the physical process is applied by a trimmer with three blades.

The Finishing Process specified by a TRIMMING SPECIFICATION for which the value of TRIM DIMENSIONS is (x, y) and the value of TRIM OFFSET is z shall be to trim the document such that the size of the document after trimming shall be x mm by y mm. The trimming operation is be applied with an offset of z mm from the Reference Edge as shown in figure 7.

The trimming Finishing Process changes the value of the *reference-size* presentation parameter to the value of TRIM DIMENSIONS. Subsequent Finishing Processes which are dependent on the value of the *reference-size* presentation parameter may be affected. The medium Reference Coordinate System shall be associated with the trimmed document in such a way that:

— one of the edges of the length x mm corresponds to the positive x-axis of the medium Reference Coordinate System;

— one of the edges of the length y mm corresponds to the positive y-axis of the medium Reference Coordinate System.

14.8 DIE CUTTING SPECIFICATION

A DIE CUTTING SPECIFICATION shall be either:

— a NAMED DIE CUTTING structure element,

— a DIE CUTTING PARAMETERS structure element.

14.8.1 NAMED DIE CUTTING

A NAMED DIE CUTTING structure element shall be a base structure element The value of a NAMED DIE CUTTING structure element shall be of type *Environment Identifier*.

The value of a NAMED DIE CUTTING structure element shall identify a die cutting operation. There are no Public Object Identifier values for NAMED DIE CUTTING defined by this International Standard.

14.8.2 DIE CUTTING PARAMETERS

A DIE CUTTING PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— one DIE CUT POSITION structure element,

— one DIE CUT NAME structure element.

The immediate subordinates may occur in any order.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

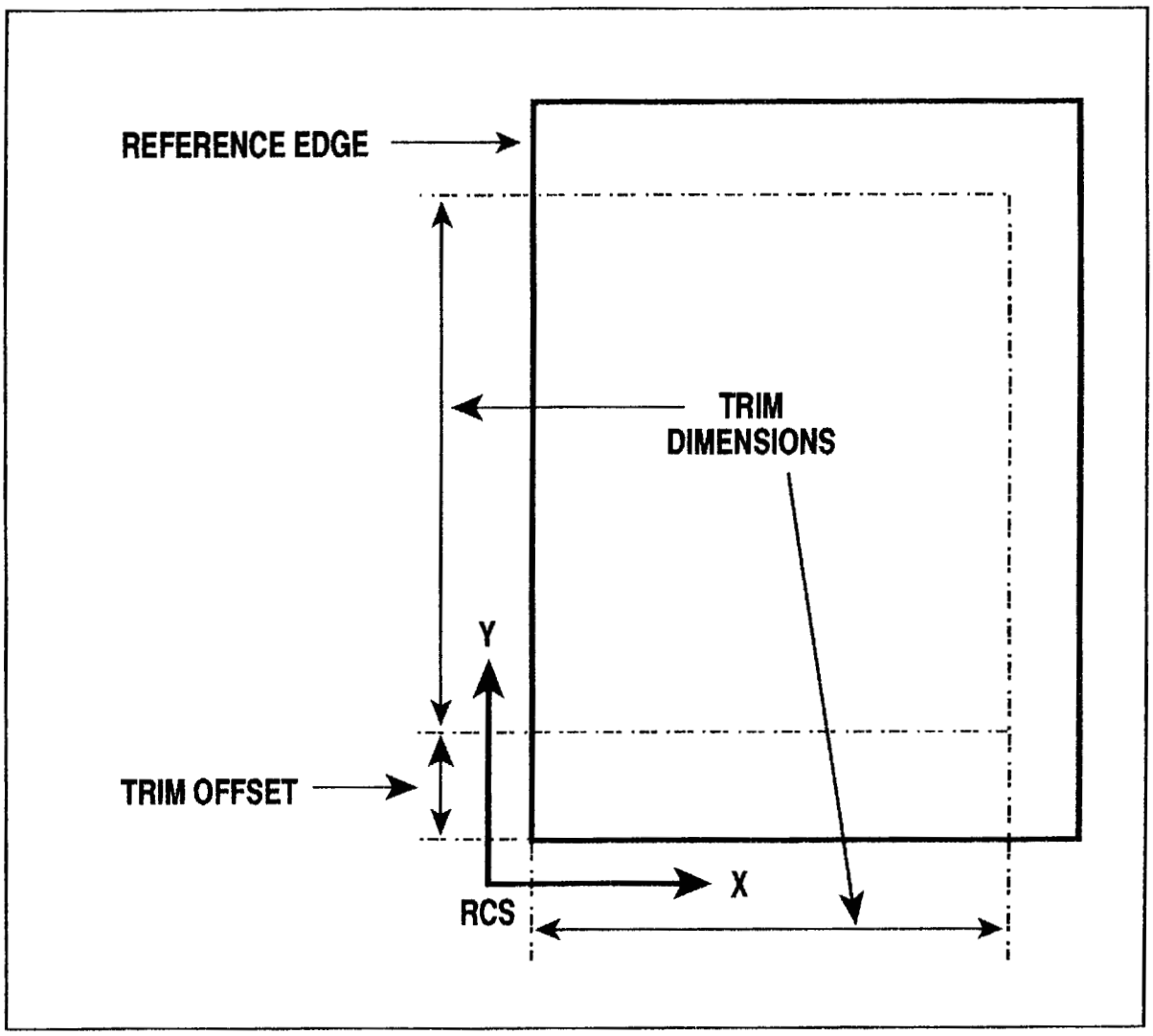

Figure 7 — Trimming Finishing Process

14.8.3 DIE CUT POSITION

A DIE CUT POSITION structure element shall be an XYDIMENSIONS structure element. There is no default value for DIE CUT POSITION.

14.8.4 DIE CUT NAME

A DIE CUT NAME structure element shall be a base structure element. The value of a DIE CUT NAME structure element shall be of type *Environment Identifier*.

The value of DIE CUT NAME specifies the name of a die cut to be applied by the die cutting Finishing Process. There are no Public Object Identifier values for DIE CUT NAME defined by this International Standard. There is no default value for DIE CUT NAME.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

14.8.5 Die Cut Finishing Process

A DIE CUTTING SPECIFICATION specifies a process which applies a die cut constrained to a predetermined and possibly arbitrary shape. A DIE CUT POSITION value of (*x*, *y*) specifies that the die used for die cutting is positioned at a location (*x*, *y*) in the medium Reference Coordinate System. The offset for the die cut relative to its actual shape is a property of the die cut itself. The Die Cutting Finishing Process specified such a DIE CUTTING SPECIFICATION is shown in figure 8.

14.9 PUNCHING SPECIFICATION

A PUNCHING SPECIFICATION shall be either:

— a NAMED PUNCHING structure element;

— a PUNCHING PARAMETERS structure element.

14.9.1 NAMED PUNCHING

A NAMED PUNCHING structure element shall be a base structure element. The value of a NAMED PUNCHING structure element shall be of type *Environment Identifier*.

The value of a NAMED PUNCHING structure element shall identify a punching operation. Public Object Identifier values for NAMED PUNCHING are defined by ISO/IEC 10175 as standard values for the attribute *punching*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

14.9.2 PUNCHING PARAMETERS

A PUNCHING PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— zero or one PROCESS OFFSET structure element;

— zero or one HEAD LOCATIONS structure element;

— zero or one PUNCH DIAMETER structure element.

The immediate subordinates may occur in any order.

The default behavior shall be to position the punched holes parallel to and near the Reference Edge.

14.9.3 PUNCH DIAMETER

A PUNCH DIAMETER structure element shall be a DIMENSION structure element. The default value for PUNCH DIAMETER shall be system dependent

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

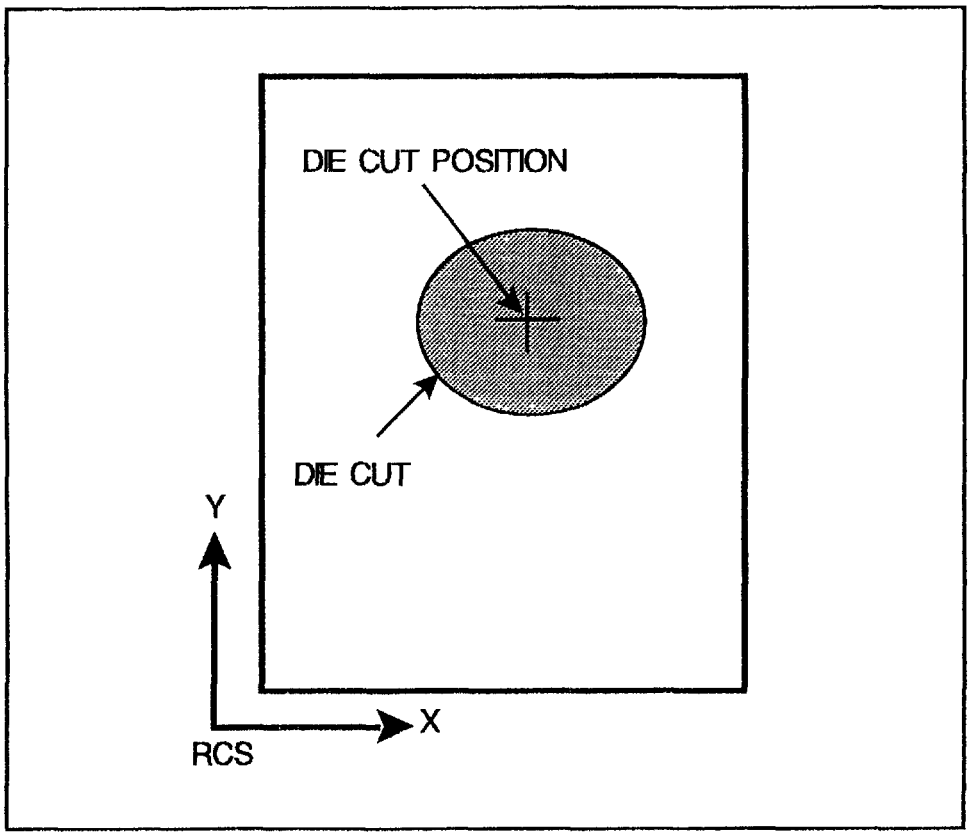

Figure 8 — Die Cutting Finishing Process

14.9.4 *Punching Finishing Process*

The Finishing Process which is specified by a PUNCHING SPECIFICATION for which the value specified by PROCESS OFFSET is *y* and the value specified by HEAD LOCATIONS is $x_1, \ldots, x_n$, and the value specified by PUNCH DIAMETER is *d*, shall punch or drill *n* holes of diameter *d* mm in the document along a Process Axis which is offset *y* mm from the Reference Edge, at locations $x_1, \ldots, x_n$ mm from the *x*-axis or *y*-axis of the medium Reference Coordinate System.

If the heads which apply the punching operation are positionable within only a limited *process-offset* range, the system shall use a default offset and a structure warning shall be generated.

If the PUNCHING SPECIFICATION specifies a *process-offset* which specifies head positioning outside of the achievable range, the system shall use a default offset and a structure warning shall be generated.

14.10 PERFORATING SPECIFICATION

A PERFORATING SPECIFICATION shall be either:

— a NAMED PERFORATION structure element;

— a PERFORATION PARAMETERS structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

14.10.1 NAMED PERFORATION

A NAMED PERFORATION structure element shall be a base structure element. The value of a NAMED PERFORATION structure element shall be of type *Environment Identifier*.

The value of a NAMED PERFORATION structure element shall identify a perforating operation. Public Object Identifier values for NAMED PERFORATION are defined by ISO/IEC 10175 as standard values for the attribute *perforation*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

14.10.2 PERFORATION PARAMETERS

A PERFORATION PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— one PROCESS OFFSET structure element;

— one HEAD LOCATIONS structure element;

— one PERFORATION TYPE structure element.

The immediate subordinates may occur in any order.

14.10.3 PERFORATION TYPE

A PERFORATION TYPE structure element shall be a base structure element. The value of a PERFORATION TYPE structure element shall be of type *Environment Identifier*.

The value of a PERFORATION TYPE structure element shall identify a type of perforation. Public Object Identifier values for PERFORATION TYPE are defined by ISO/IEC 10175 as standard values for the attribute *perforation-type*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

The default value of PERFORATION TYPE shall be the Public Object Identifier value which corresponds to the Object Name *dpi::perforation::type::default*.

14.10.4 Perforating Finishing Process

A PERFORATING SPECIFICATION specifies a Finishing Process which applies one or more runs of perforations perpendicular to the Reference Edge. The number of perforations along a run is system dependent. The length of the run of perforations is the length of the edge perpendicular to the Reference Edge. The heads which apply perforations are positioned along the Reference Edge using the HEAD LOCATIONS value.

The Finishing Process which is specified by a PERFORATING SPECIFICATION for which the value specified by HEAD LOCATIONS is $x_1, \ldots, x_n$, shall apply n runs of perforations in the document perpendicular to the Process Axis at locations $x_1, \quad, x_n$ mm from the x-axis or y-axis of the medium Reference Coordinate System.

If the heads which apply the perforating operation are positionable within only a limited range, the system shall use default locations and a structure warning shall be generated.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

14.11 SLITTING SPECIFICATION

A SLITTING SPECIFICATION shall be either:

— a NAMED SLITTING structure element;

— a SLITTING PARAMETERS structure element.

14.11.1 NAMED SLITTING

A NAMED SLITTING structure element shall be a base structure element. The value of a NAMED SLITTING structure element shall be of type *Environment Identifier*.

The value of a NAMED SLITTING structure element shall identify a slitting operation. Public Object Identifier values for NAMED SLITTING are defined by ISO/IEC 10175 as standard values for the attribute *slitting*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

14.11.2 SLITTING PARAMETERS

A SLITTING PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— one HEAD LOCATIONS structure element.

The immediate subordinates may occur in any order.

14.11.3 Slitting Finishing Process

A SLITTING SPECIFICATION specifies a Finishing Process which cuts the set of sheets being finished perpendicular to the Reference Edge in one or more places. The length of a slit is the length of the edge perpendicular to the Reference Edge. The heads which apply slitting are positioned along the Reference Edge using the *head-locations* values.

The Finishing Process which is specified by a SLITTING SPECIFICATION for which the value specified by HEAD LOCATIONS is $x_1, \ldots, x_n$, shall cut n slits in the document perpendicular to the Process Axis at locations $x_1, \ldots, x_n$ mm from the x-axis or y-axis of the medium Reference Coordinate System.

If the heads which apply the slitting operation are positionable within only a limited range, the system shall use default locations and a structure warning shall be generated.

14.12 INSERT SPECIFICATION

An INSERT SPECIFICATION shall be either:

— a NAMED INSERTING structure element;

— an INSERTING PARAMETERS structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642606 139

14.12.1 NAMED INSERTING

A NAMED INSERTING structure element shall be a base structure element. The value of a NAMED INSERTING structure element shall be of type *Environment Identifier*.

The value of a NAMED INSERTING structure element shall identify an inserting operation. There are no Public Object Identifier values for NAMED INSERTING defined by this International Standard.

14.12.2 INSERTING PARAMETERS

An INSERTING PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— one INSERT SHEET LIST structure elements.

The immediate subordinates may occur in any order.

14.12.3 INSERT SHEET LIST

The INSERT SHEET LIST structure element shall be a composite structure element which shall have the the following immediate subordinates:

— one or more INSERT SHEET structure elements.

Each INSERT SHEET structure element specifies a set of properties which identify a sheet to be inserted in the document by an inserting Finishing Process.

14.12.4 INSERT SHEET

An INSERT SHEET structure element shall be a composite structure element which shall have the the following immediate subordinates:

— one INSERT IDENTIFIER structure element;

— zero or one INSERT TOP SURFACE structure element;

— one INSERT AFTER structure element;

— zero or one INSERT MESSAGE structure element.

These structure elements may occur in any order.

14.12.5 INSERT IDENTIFIER

An INSERT IDENTIFIER structure element shall be one of:

— an INSERT NAME structure element;

— an INSERT BIN structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642607 075

14.12.6 INSERT NAME

An INSERT NAME structure element shall be a base structure element. The value of an INSERT NAME structure element shall be of type *Environment Identifier*.

The value of an INSERT NAME structure element shall identify an insert sheet to be inserted into the document. There are no Public Object Identifier values for INSERT NAME defined by this International Standard.

14.12.7 INSERT BIN

An INSERT BIN structure element shall be a base structure element. The value of an INSERT BIN structure element shall be of type *Cardinal*.

A value of INSERT BIN shall identify a bin from which the insert sheet can be obtained by the inserting Finishing Process. The mapping from Cardinal values (bin numbers) to physical bins is system dependent.

INSERT BIN should be used only locally and in conjunction with an INSERT MESSAGE or DOCUMENT START MESSAGE which specifies to the operator the insert sheet which is to be loaded into the specified insert bin.

14.12.8 INSERT TOP SURFACE

An INSERT TOP SURFACE structure element shall be a base structure element. The type of the value of an INSERT TOP SURFACE structure element shall be an Enumerated type which identifies the insert sheet surface which shall face in the same direction as the normal reading order of the other sheets in the associated document. The names of the values of this type and the insert top surface identified shall be as shown in table 21.

Table 21

Value	Meaning
top	The normal top surface of the insert sheet
bottom	The normal bottom surface of the insert sheet.

The default value of INSERT TOP SURFACE shall be *top*.

14.12.9 INSERT AFTER

An INSERT AFTER structure element shall be a base structure element. The value of an INSERT AFTER structure element shall be of type *Cardinal*.

The value of an INSERT AFTER structure element shall identify the ordinal number of the PAGE of the document after which the insert sheet should be inserted. If multiple PAGE images are placed on a single instance of the medium, the insert sheet shall be inserted after the instance of the medium on which the designated PAGE image has been printed.

14.12.10 INSERT MESSAGE

An INSERT MESSAGE specifies a message which may describe the insert sheet, the suggested fallback action, or provide other information. It is intended that this message be delivered to an operator prior to and as close as possible to the finishing of the document. The INSERT MESSAGE may effect printing management, but any such effect shall be system dependent. The INSERT MESSAGE may also be ignored.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

An INSERT MESSAGE structure element shall be a base structure element. The value of an INSERT MESSAGE shall be of type *Printable String*.

14.12.11 Inserting Finishing Process

An INSERT SPECIFICATION specifies a Finishing Process which inserts insert sheets into a collated set of imaged PAGEs. All insert sheets must have a "normal orientation", and identifiable top, bottom, left and right edges and top and bottom surfaces. Each insert sheet is introduced into a collated set of printed sheets created by the marking device after the marking phase has completed for the physical sheet preceding the preprinted insert. Inserting mechanisms may insert sheets after all pages in the job have been marked.

A sheet insertion process uses an inserting mechanism which is generally provided with the capability to insert sheets from one or more of several input bins. Insert sheets are not identified through the Medium Document Production, and are not intended to be input to the finishing process from input bins designed for the marking engine proper.

14.13 COVERS SPECIFICATION

A COVERS SPECIFICATION shall be either;

— a NAMED COVERS structure element;

— a COVERS PARAMETERS structure element.

14.13.1 NAMED COVERS

A NAMED COVERS structure element shall be a base structure element. The value of a NAMED COVERS structure element shall be of type *Environment Identifier*.

The value of a NAMED COVERS structure element shall identify a set of covers to be applied to a document. Public Object Identifier values for NAMED COVERS are defined by ISO/IEC 10175 as standard values for the attribute *covers*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

14.13.2 COVERS PARAMETERS

A COVERS PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element,

— zero or one FRONT COVER structure element;

— zero or one BACK COVER structure element.

The immediate subordinates may occur in any order.

14.13.3 FRONT COVER

A FRONT COVER structure element shall be either.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642609 948

— a COVER NAME structure element;

— a MEDIUM SPECIFICATION structure element.

14.13.4 BACK COVER

A BACK COVER structure element shall be either:

— a COVER NAME structure element;

— a MEDIUM SPECIFICATION structure element.

14.13.5 COVER NAME

A COVER NAME structure element shall be a base structure element. The value of a COVER NAME structure element shall be of type *Environment Identifier*.

The value of a COVER NAME structure element shall identify a cover to be added to a document. Public Object Identifier values for COVER NAME are defined by ISO/IEC 10175 as standard values for the attribute *cover-name*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

14.13.6 Covers Finishing Process

A COVERS SPECIFICATION specifies a Finishing Process that adds a front cover, a back cover, or both to the document.

14.14 FOLDING SPECIFICATION

A FOLDING SPECIFICATION shall be either:

— a NAMED FOLDING structure element;

— a FOLDING PARAMETERS structure element.

14.14.1 NAMED FOLDING

A NAMED FOLDING structure element shall be a base structure element. The value of a NAMED FOLDING structure element shall be of type *Environment Identifier*.

The value of a NAMED FOLDING structure element shall identify a folding operation. Public Object Identifier values for NAMED FOLDING are defined by ISO/IEC 10175 as standard values for the attribute *folding*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative).

14.14.2 FOLDING PARAMETERS

A FOLDING PARAMETERS structure element shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642610 66T

— zero or one HEAD LOCATIONS structure element.

The immediate subordinates may occur in any order.

14.14.3 Folding Finishing Process

A FOLDING SPECIFICATION specifies a Finishing Process which applies one or more folds to the document. The folds are oriented perpendicular to the Reference Edge. The heads which apply folding are positioned along the Reference Edge using the Locations value.

The folding Finishing Process changes the value of the *reference-size* presentation parameter. Subsequent operations which are dependent on the value of the *reference-size* presentation parameter may be affected.

If the FOLDING SPECIFICATION specifies a *process-offset* which specifies head positioning outside of the achievable range, the system shall use a default offset and a structure warning shall be generated.

14.15 OTHER FINISHING SPECIFICATION

The Finishing Document Production Instruction syntax allows the inclusion of additional Finishing Specifications by means of OTHER FINISHING SPECIFICATION structure elements. Each OTHER FINISHING SPECIFICATION consists of Finishing Process name and a set of parameters which specify the Finishing Process. The set of parameters depends on the Finishing Process being specified. The OTHER FINISHING SPECIFICATION syntax allows for the specification of Finishing Process parameters either by a NAMED FINISHING structure element or by a parameter list similar to those used for the finishing specifications defined by this International Standard.

An OTHER FINISHING SPECIFICATION shall be a composite structure element which shall have the the following immediate subordinates:

— one FINISHING OP NAME structure element;

— zero or one FINISHING OP DETAILS structure element.

14.15.1 FINISHING OP NAME

A FINISHING OP NAME structure element shall be a base structure element. The value of a FINISHING OP NAME structure element shall be of type *Public Object Identifier*.

The value of a FINISHING OP NAME structure element shall identify an Additional Finishing Specification. There are no Public Object Identifier values for FINISHING OP NAME structure elements defined by this International Standard.

14.15.2 FINISHING OP PARAMETERS

A FINISHING OP PARAMETERS structure element shall be either:

— a NAMED FINISHING OP PARAMETERS structure element;

— a SPECIFIED FINISHING OP PARAMETERS structure element.

14.15.3 NAMED FINISHING OP PARAMETERS

A NAMED FINISHING OP PARAMETERS structure element shall be a base structure element. The value of a NAMED FINISHING OP PARAMETERS structure element shall be of type *Environment Identifier*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642611 5T6

14.15.4 SPECIFIED FINISHING OP PARAMETERS

A SPECIFIED FINISHING OP PARAMETERS shall be a composite structure element which shall have the the following immediate subordinates:

— zero or one REFERENCE SIZE structure element;

— zero or one REFERENCE EDGE structure element;

— zero or one JOG EDGE structure element;

— zero or one PROCESS OFFSET structure element;

— zero or one HEAD LOCATIONS structure element;

— zero or one ADDITIONAL PARAMETERS structure element.

The immediate subordinates may occur in any order.

14.15.5 ADDITIONAL PARAMETERS

An ADDITIONAL PARAMETERS structure element may be a base structure element or a composite structure element, depending on the value of the FINISHING OP NAME of the most immediately superior OTHER FINISHING SPECIFICATION.

14.15.6 OTHER FINISHING SPECIFICATION Finishing Processes

The specification of the Finishing Process specified by an OTHER FINISHING SPECIFICATION shall be the responsibility of the owner of the Public Object Identifier which is the value of the FINISHING OP NAME. The effect of an OTHER FINISHING SPECIFICATION shall be either:

— the Finishing Process or fallback for the Finishing Process specified by the owner of the Public Object Identifier;

— that the OTHER FINISHING SPECIFICATION is ignored.

The definition of the syntax of the FINISHING OP DETAILS structure element to a particular FINISHING OP NAME value, and the Finishing Process specified by the OTHER FINISHING SPECIFICATION are outside of the scope of this International Standard.

15 Management Document Production Instructions

Management Document Production Instructions affect the management of the document production process. The presentation parameters which are affected by the Management Document Production Instructions defined by this International Standard, and the aspect of document presentation which is affected by each of them is shown in table 22.

Table 22

Name	Affects
current-colorant-set	colorant set for pages in a BLOCK
timeout	maximum time for interpretation and presentation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642612 432

Name	Affects
abort-policy	effect of any errors or warnings
start-page-type	type of start page or break page
separator-page-type	type of separator page
end-page-type	type of end page
output-bin	bin or bins to which output is to be delivered
output-specification	output specification

15.1 Colorant Set Document Production Instruction

The Colorant Set Document Production Instruction identifies a set of physical colorants to be used during the presentation of the SPDL document. Physical colorants used to render the document are distinct from primaries in Color Space Resource Specifications. Specifically, there is no direct relationship between primaries specified in a Color Space Resource Specification and physical colorants, and the mapping of color specifications to device output is device dependent.

A colorant set can be specified by a colorant set name or by a set of individual colorant specifications. The way in which a system selects a physical colorant set is system dependent. Systems may in fact select a single default colorant set for any Colorant Set Document Production Instruction.

NOTES

61 The rendering of the SPDL document according to the mechanisms provided for by a particular Color Space may be facilitated if appropriate physical colorants are available.

62 The gamut of colors available to the Presentation Process, and hence the final appearance of the document, may be affected by the choice of physical colorants. An example of this is for highlight color printing, when the marking engine uses only a black colorant and a single highlight color colorant.

15.1.1 COLORANT SET DPI

The Colorant Set Document Production Instruction is represented by a COLORANT SET DPI structure element. A COLORANT SET DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

– zero or one COLORANT SET MESSAGE structure element;

– one COLORANT SET SPECIFICATION structure element.

The immediate subordinates shall occur in the specified order

15.1.1.1 COLORANT SET MESSAGE

A COLORANT SET MESSAGE structure element shall be a base structure element. The value of a COLORANT SET MESSAGE structure element shall be of type *Printable String*.

The COLORANT SET MESSAGE specifies a message which may describe the specified colorant set, the suggested fallback action, or provide other information. It is intended that this message be delivered to an operator prior to and as close as possible to the printing of the document

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

15.1.1.2 COLORANT SET SPECIFICATION

A COLORANT SET SPECIFICATION structure element shall be either:

— a COLORANT SET IDENTIFIER structure element;

— a COLORANT SET DESCRIPTION structure element.

15.1.1.3 COLORANT SET IDENTIFIER

A COLORANT SET IDENTIFIER structure element shall be a base structure element. The value of the COLORANT SET IDENTIFIER structure element shall be of type *Environment Identifier*.

15.1.1.4 COLORANT SET DESCRIPTION

A COLORANT SET DESCRIPTION structure element shall be a composite structure element which shall have the following immediate subordinates:

— one COLORANT SET CLASS IDENTIFIER structure element;

— one or more COLORANT IDENTIFIER structure elements.

The immediate subordinates shall occur in the specified order.

15.1.1.5 COLORANT SET CLASS

A COLORANT SET CLASS IDENTIFIER structure element shall be a base structure element. The value of a COLORANT SET CLASS IDENTIFIER structure element shall be of type *Public Object Identifier*. The Object Names for which Public Object Identifiers are defined by this International Standard and the meaning of those values shall be as shown in table 23.

Table 23

Object Name	Meaning
dpi::colorant-set-class::cmy	cyan, magenta, and yellow subtractive colorants
dpi::colorant-set-class::cmyk	cyan, magenta, yellow, and black subtractive colorants
dpi::colorant-set-class::rgb	red, green, and blue additive colorants
dpi::colorant-set-class::kx	two color subtractive colorants
dpi::colorant-set-class::monochrome	one colorant

The value of a COLORANT SET CLASS IDENTIFIER structure element shall identify the class of the colorant set.

15.1.1.6 COLORANT IDENTIFIER

A COLORANT IDENTIFIER structure element shall be a base structure element. The value of a COLORANT IDENTIFIER structure element shall be of type *Environment Identifier*.

The value of each COLORANT IDENTIFIER structure element shall identify one of the colorants in the colorant set. There are no Public Object Identifier values for COLORANT IDENTIFIER structure elements defined by this International Standard.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642614 205

15.1.2 Colorant Set Document Production Instruction Processing

The effect of the Colorant Set Document Production Instruction shall be to set the value of the *current-colorant-set* presentation parameter to the colorant set specified by the COLORANT SET SPECIFICATION for the scope of the most immediately superior DPI DECLARATION. A Colorant Set Document Production Instruction may affect printing management; the effects of a Colorant Set Document Production Instruction on printing management (if any) shall be system dependent.

15.1.3 Colorant Set Document Production Instruction Reconciliation

The Colorant Set Document Production Instruction is meaningful in any PAGE or PAGESET, or in Supplementary DPI. Any Colorant Set Document Production Instruction in the PROLOGUE of a PICTURE which is not a PAGE shall be ignored If Colorant Set Document Production Instructions occur in multiple PAGEs or PAGESETs within a document, the Colorant Set Document Production Instruction in the subordinate BLOCK shall take precedence for its scope. If Colorant Set Document Production Instructions occur in Supplementary DPI and Document DPI, the Supplementary DPI shall take precedence over the Document DPI.

15.1.4 Colorant Set Document Production Instruction Fallback

Since the method of assigning a colorant set to a COLORANT SET SPECIFICATION is system dependent, no fallback is required.

15.2 Document Start Message Document Production Instruction

The Document Start Message Document Production Instruction specifies a Printable String which is to be displayed on the operator interface (if any) and acknowledged before the document is printed, as close to the time of printing as possible.

15.2.1 DOCUMENT START MESSAGE DPI

The Document Start Message Document Production Instruction shall be represented by a DOCUMENT START MESSAGE DPI structure element.

A DOCUMENT START MESSAGE DPI structure element shall be a base structure element. The value of a DOCUMENT START MESSAGE DPI structure element shall be of type *Printable String*.

15.2.2 Document Start Message Document Production Instruction Processing

The effect of a Document Start Message Document Production Instruction shall be that the Printable String which is the value of the DOCUMENT START MESSAGE DPI shall be displayed on the operator interface (if any) and acknowledged before the document is printed, as close to the time of printing as possible.

15.2.3 Document Start Message Document Production Instruction Reconciliation

Document Start Message Document Production Instructions are meaningful only in a DOCUMENT or in Supplementary DPI. If a Document Start Message Document Production Instruction occurs as a Supplementary DPI, the Supplementary DPI shall take precedence over the Document DPI.

15.2.4 Document Start Message Document Production Instruction Fallback

Systems which cannot display a Document Start Message shall indicate failure to do so by generating a structure warning.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642615 141

15.3 Auxiliary Page Type Document Production Instruction

Printers may print one or more special page called "auxiliary pages" to identify a document or a set of pages. There are three types of auxiliary pages:

— a start page, which is printed before the first page of the first copy of the document;

— separator pages, which are printed between "page sets" (determined as specified below);

— an end page which is printed after the last page of the last copy of the document.

The start page usually contains a distinctive graphic design to enable rapid recognition of document boundaries, and identifies the print job, the document, and the print job recipient. It may also give the document name, creation date, printing date, recipient name, and messages describing errors, etc.. Information printed on the start page may also be supplied by external sources.

Separator pages are usually unprinted, but are commonly of a distinctive color (slip pages). The set of pages which constitutes a page set depends on how the document is being printed:

— If multiple copies are being printed in collated form, a page set consists of a single copy of the document.

— If multiple copies are being printed in uncollated form (all copies of the first page, followed by all copies of the second page, etc.) a page set consists of all copies of a single page.

— If multiple copies are being printed in block collated form (all copies of pages *1* through *n* in collated form, followed by all copies of pages $n+1$ through *m* in collated form, etc) a page set consists of a single copy of a set of pages (*1* through *n*, $n+1$ through *m*, etc.).

The end page may also contain a distinctive graphic design or be of a distinctive color, and may contain messages describing errors, etc. which may not be available at the time at which the start page is generated.

The format and contents of auxiliary pages are system dependent. The Auxiliary Page Type Document Production Instruction provides control over the characteristics of the auxiliary pages desired.

15.3.1 AUXILIARY PAGE TYPE DPI

The Auxiliary Page Type Document Production Instruction shall be represented by a AUXILIARY PAGE TYPE DPI structure element. An AUXILIARY PAGE TYPE DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one START PAGE TYPE structure element;

— zero or one SEPARATOR PAGE TYPE structure element;

— zero or one END PAGE TYPE structure element.

The immediate subordinates may occur in any order.

15.3.2 START PAGE TYPE

A START PAGE TYPE structure element shall be a base structure element. The value of a START PAGE TYPE structure element shall be of type *Environment Identifier*.

The value of a START PAGE TYPE structure element identifies the type of start page to be provided. Public Object Identifier values for START PAGE TYPE are defined by ISO/IEC 10175 as standard values for the attribute

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642616 088

auxiliary-sheet or as the generic Document Production Instruction value *dpi::generic::none*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative). The specific formats and contents of the start pages which are provided as instances of the types of auxiliary pages identified by these Public Object Identifiers are system dependent.

15.3.3 SEPARATOR PAGE TYPE

A SEPARATOR PAGE TYPE structure element shall be a base structure element. The value of a SEPARATOR PAGE TYPE structure element shall be of type *Environment Identifier*.

The value of a SEPARATOR PAGE TYPE structure element identifies the type of separator page to be provided. Public Object Identifier values for SEPARATOR PAGE TYPE are defined by ISO/IEC 10175 as standard values for the attribute *auxiliary-sheet* or as the generic Document Production Instruction value *dpi::generic::none*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative). The specific formats and contents of the separator pages which are provided as instances of the types of auxiliary pages identified by these Public Object Identifiers are system dependent.

15.3.4 END PAGE TYPE

An END PAGE TYPE structure element shall be a base structure element. The value of an END PAGE TYPE structure element shall be of type *Environment Identifier*.

The value of an END PAGE TYPE structure element identifies the type of end page to be provided. Public Object Identifier values for END PAGE TYPE are defined by ISO/IEC 10175 as standard values for the attribute *auxiliary-sheet* or as the generic Document Production Instruction value *dpi::generic::none*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative). The specific formats and contents of the end pages which are provided as instances of the types of auxiliary pages identified by these Public Object Identifiers are system dependent.

15.3.5 Auxiliary Page Type Document Production Instruction Processing

The effect of the Auxiliary Page Type Document Production Instruction shall be to set the value of the *start-page-type*, *separator-page-type* and *end-page-type* presentation parameters as follows:

a) the value of the *start-page-type* presentation parameter is set to the value of START PAGE TYPE if present, or to a system dependent value if no START PAGE TYPE is present;

b) the value of the *separator-page-type* presentation parameter is set to the value of SEPARATOR PAGE TYPE if present, or to a system dependent value if no SEPARATOR PAGE TYPE is present;

c) the value of the *end-page-type* presentation parameter is set to the value of END PAGE TYPE if present, or to a system dependent value if no END PAGE TYPE is present

The default value of the *start-page-type*, *separator-page-type* and *end-page-type* presentation parameters shall be system dependent.

15.3.6 Auxiliary Page Type Document Production Instruction Reconciliation

The Auxiliary Page Type Document Production Instruction is meaningful only in a DOCUMENT or in Supplementary DPI. An Auxiliary Page Type Document Production Instruction which occurs in a BLOCK which is not a DOCUMENT shall be ignored

If a Auxiliary Page Type Document Production Instruction occurs as a Supplementary DPI, the Supplementary DPI shall take precedence over any Document DPI.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

15.3.7 Auxiliary Page Type Document Production Instruction Fallback

The printer may override any of the values specified for the values of the the *start-page-type*, *separator-page-type* and *end-page-type* presentation parameters by an Auxiliary Page Type Document Production Instruction. Auxiliary Page Type Document Production Instructions may be ignored on soft copy devices With the exception of soft copy devices, a Presentation Process which does not provide a specified auxiliary page of the specified type shall raise a structure warning.

15.4 Document Comment Document Production Instruction

The Document Comment Document Production Instruction specifies a Printable String which is printed on the break page (if any).

15.4.1 DOCUMENT COMMENT DPI

The Document Comment Document Production Instruction is represented by a DOCUMENT COMMENT DPI structure element. A DOCUMENT COMMENT DPI structure element shall be a base structure element. The value of a DOCUMENT COMMENT DPI structure element shall be of type *Printable String*.

15.4.2 Document Comment Document Production Instruction Processing

The effect of a Document Comment Document Production Instruction shall be that the Printable String which is the value of the DOCUMENT COMMENT DPI shall be printed on the break page (if any).

15.4.3 Document Comment Document Production Instruction Reconciliation

Document Comment Document Production Instructions are meaningful only in a DOCUMENT. If a Document Comment Document Production Instruction occurs as a Supplementary DPI, the Supplementary DPI shall take precedence over the Document DPI.

15.4.4 Document Comment Document Production Instruction Fallback

Systems which do not print a break page shall ignore the Document Comment Document Production Instruction. Systems which cannot print the entire Printable String on the break page shall print as much of the Printable String as possible.

15.5 Timeout Document Production Instruction

The Timeout Document Production Instruction specifies the maximum time which will be allowed for the actual interpretation and presentation of the document.

15.5.1 TIMEOUT DPI

The Timeout Document Production Instruction shall be represented by a TIMEOUT DPI structure element. A TIMEOUT DPI structure element shall be a base structure element. The value of a TIMEOUT DPI structure element shall be of type *Cardinal*.

15.5.2 Timeout Document Production Instruction Processing

The processing of the Timeout Document Production Instruction consists of setting the value of the *timeout* presentation parameter to the value of the TIMEOUT DPI.

The *timeout* presentation parameter specifies the maximum time (in seconds) which will be allowed for the actual interpretation and presentation of the document. The time begins when interpretation of the document

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642618 950

for presentation begins, and may be extended by the Presentation Process if other tasks are being performed concurrently. If this time is exceeded document printing shall be aborted unless either:

— the value of timeout is zero, indicating that the timeout function is to be disabled;

— presentation is on an interactive soft copy device or other device for which timeout is not appropriate.

The default value for the *timeout* presentation parameter is system dependent, but shall not be less than 360.

15.5.3 Timeout Document Production Instruction Reconciliation

The Timeout Document Production Instruction is meaningful only in a DOCUMENT or in Supplementary DPI.

If a Timeout Document Production Instruction occurs as a Supplementary DPI, the Supplementary DPI shall take precedence over the Document DPI.

15.5.4 Timeout Document Production Instruction Fallback

The default value for the *timeout* presentation parameter is system dependent. A system may restrict timeout values, provided that values of at least 360 are allowed.

15.6 Document Abort Policy Document Production Instruction

The Document Abort Policy Document Production Instruction specifies the effect of errors and exceptions which occur during the presentation of the document Such errors and exceptions are classified as either *structure warnings* or *structure errors.*

15.6.1 ABORT POLICY DPI

The Document Abort Policy Document Production Instruction shall be represented by an ABORT POLICY DPI structure element

An ABORT POLICY DPI structure element shall be a base structure element. The value of an ABORT POLICY DPI structure element shall be of type *Environment Identifier*. The Object Names for which Public Object Identifiers are defined by this International Standard and the meaning of those values shall be as shown in table 24.

Table 24

Object Name	Meaning
dpi::abort-policy::on-warning	document presentation shall be aborted by the occurrence of any structure error or structure warning
dpi::abort-policy::on-error	document presentation shall be aborted by the occurrence of any structure error, but not by any structure warning
dpi.:abort-policy.:struggle-on	the Presentation Process shall provide its best effort to present the document even in the presence of structure errors or structure warnings.

15.6.2 Document Abort Policy Document Production Instruction Processing

The effect of the Document Abort Policy Document Production Instruction is to set the value of the *abort-policy* parameter to the value of the Abort Policy Document Production Instruction. The *abort-policy* parameter iden-

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642619 897

tifies the conditions under which document presentation is to be aborted The default value of *abort-policy* parameter shall be system dependent.

All systems shall recognize the values of *abort-policy* specified by this International Standard and implement the associated policies. Systems may also recognize other values for the ABORT POLICY DPI and implement policies which are defined by the owners of those values

15.6.3 Document Abort Policy Document Production Instruction Reconciliation

Document Abort Policy Document Production Instructions are meaningful only in a DOCUMENT or in Supplementary DPI. If a Document Abort Policy Document Production Instruction occurs as a Supplementary DPI, the Supplementary DPI shall take precedence over the Document DPI

15.6.4 Document Abort Policy Document Production Instruction Fallback

Systems may recognize other values for the ABORT POLICY DPI and implement policies which are defined by the owners of those values. The effect of an ABORT POLICY DPI which specifies a value which is not supported shall be that the default value is used.

15.7 Output Bin Document Production Instruction

The Output Bin Document Production Instruction specifies the bin or bins to which to which the completed document is to be delivered.

15.7.1 OUTPUT BIN DPI

The Output Bin Document Production Instruction shall be represented by an OUTPUT BIN DPI structure element. An OUTPUT BIN DPI structure element shall be a composite structure element which shall have the following immediate subordinates:

— zero or one OUTPUT BIN NAME or OUTPUT BIN NUMBER structure element;

— zero or one OUTPUT POSITIONS structure elements.

The immediate subordinates may occur in any order.

15.7.1.1 OUTPUT BIN NAME

A OUTPUT BIN NAME structure element shall be a base structure element. The value of a OUTPUT BIN NAME structure element shall be of type *Environment Identifier.*

A OUTPUT BIN NAME structure element specifies a single output tray or bin or a set of output trays or bins. Public Object Identifier values for the OUTPUT BIN NAME structure element are defined by ISO/IEC 10175 as standard values for the attribute *output-bin*. The Public Object Identifier values defined by ISO/IEC 10175 and the meaning of those values are described in annex F (informative). The mapping of OUTPUT BIN NAME values to physical output trays or bins is system dependent.

15.7.1.2 OUTPUT BIN NUMBER

An OUTPUT BIN NUMBER structure element shall be base structure element. The value of an OUTPUT BIN NUMBER structure element shall be of type *Cardinal.*

An OUTPUT BIN NUMBER structure element specifies a single output tray or bin or a set of output trays or bins. The mapping of OUTPUT BIN NUMBER values to physical output trays or bins is system dependent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642620 509

15.7.1.3 OUTPUT POSITIONS

An OUTPUT POSITIONS structure element shall be base structure element. The value of an OUTPUT POSITIONS structure element shall be of type *Cardinal*.

The OUTPUT POSITIONS structure element specifies the number of output trays or bins which are to be used. The mapping of output positions to physical output trays or bins and the sequence of output positions used is system dependent

NOTE 63 Systems which cannot provide the number of distinct output positions specified by Output Position Document Production Instruction may map multiple output positions to the same output tray or bin. The number of output positions actually available need not exceed one.

15.7.2 Output Bin Document Production Instruction Processing

The effect of the Output Bin Document Production Instruction shall be to set the value of the *output-bins* presentation parameter to the set of output bins or trays specified by the OUTPUT BIN DPI structure element. The *output-bins* presentation parameter specifies how the printer shall deliver the output document(s). The default value of the *output-bins* presentation parameter is system dependent.

15.7.3 Output Bin Document Production Instruction Reconciliation

An Output Bin Document Production Instruction is meaningful only in a DOCUMENT. Any Output Document Production Instruction which occurs in a BLOCK which is not a DOCUMENT shall be ignored.

If an Output Bin Document Production Instruction in Supplementary DPI, the Supplementary DPI shall take precedence over any Document DPI.

15.7.4 Output Bin Document Production Instruction Fallback

A system which cannot perform an output specification specified by the Output Bin Document Production Instruction shall generate a structure warning. If the value of the *abort-policy* presentation parameter allows for continued processing, the effect of that continued processing shall be system dependent.

15.8 Output Specification Document Production Instruction

The Output Specification Document Production Instruction specifies how printed document(s) should be output.

15.8.1 OUTPUT SPECIFICATION DPI

The Output Specification Document Production Instruction shall be represented by an OUTPUT SPECIFICATION DPI structure element. An OUTPUT SPECIFICATION DPI structure element shall be a composite structure element which shall have the following immediate subordinates

— zero or one COLLATED structure element;

— zero or one OFFSET structure element;

— zero or one BURST structure element;

— zero or more ADDITIONAL OUTPUT structure element

The immediate subordinates may occur in any order.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

15.8.2 COLLATED

The value of a COLLATED structure element specifies whether output is delivered as a collection of all copies of each page (uncollated), or as a collection of pages of each copy (collated).

A COLLATED structure element shall be a base structure element. The value of a COLLATED structure element shall be of type *Boolean*. The meaning of the values shall be as shown in table 25.

Table 25

Value	Meaning
true	the output is delivered in collated order
false	the output is delivered in uncollated order

The default value for COLLATED shall be true.

15.8.3 OFFSET

The OFFSET structure element enables control of whether multiple copies should be edge displaced relative to each other as they are stacked.

An OFFSET structure element shall be a base structure element. The value of an OFFSET structure element shall be of type *Boolean*. The meaning of the values shall be as shown in table 25.

Table 26

Value	Meaning
true	each copy is distinguishably offset in a system dependent manner from its neighbor copies
false	output is not offset stacked.

The default value for OFFSET shall be false.

15.8.4 BURST

The BURST structure element enables control of whether output on continuous media should be burst (separated into individual sheets, generally by bursting along perforations) before delivery.

A BURST structure element shall be a base structure element. The value of the BURST structure element shall be of type *Boolean*. The meaning of the values shall be as shown in table 27.

Table 27

Value	Meaning
true	output on continuous media shall be separated into individual sheets, generally by bursting along perforations
false	output on continuous media shall not be separated into individual sheets.

The default value for BURST shall be false.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642622 381

15.8.5 ADDITIONAL OUTPUT

The ADDITIONAL OUTPUT structure element enables additional output specifications. The effect of such ADDITIONAL OUTPUT structure elements is system dependent.

An ADDITIONAL OUTPUT structure element shall be a composite structure element which shall have the following immediate subordinates:

— one OUTPUT SPECIFICATION NAME structure element;

— zero or one OUTPUT SPECIFICATION VALUE structure elements.

The immediate subordinates may occur in any order.

15.8.5.1 OUTPUT SPECIFICATION NAME

A OUTPUT SPECIFICATION NAME structure element shall be a base structure element. The value of a OUTPUT SPECIFICATION NAME structure element shall be of type *Public Object Identifier*.

15.8.5.2 OUTPUT SPECIFICATION VALUE

A OUTPUT SPECIFICATION VALUE structure element may be either a base structure element or a composite structure element. The OUTPUT SPECIFICATION VALUE structure element shall depend on the value of the OUTPUT SPECIFICATION NAME structure element

15.8.5.3 OUTPUT SPECIFICATION Semantics

The specification of the semantics of an ADDITIONAL OUTPUT shall be the responsibility of the owner of the Public Object Identifier which is the value of the OUTPUT SPECIFICATION NAME. The effect of an ADDITIONAL OUTPUT on the presentation of an SPDL document shall be either:

— an effect defined by the semantics of the ADDITIONAL OUTPUT;

— that the ADDITIONAL OUTPUT is ignored

The definition of the syntax of the OUTPUT SPECIFICATION VALUE corresponding to any particular OUTPUT SPECIFICATION NAME value, and the effect of ADDITIONAL OUTPUT on the presentation of SPDL documents are outside of the scope of this International Standard.

15.8.6 Output Specification Document Production Instruction Processing

The effect of the Output Specification Document Production Instruction shall be to set the value of the *output-specification* presentation parameter to the set of output properties specified by the immediate subordinates of the OUTPUT SPECIFICATION DPI structure element. The *output-specification* presentation parameter specifies how the printer shall deliver the output document(s). The default value of the *output-specification* presentation parameter is a null list, which specifies that a system dependent output method is to be used.

15.8.7 Output Specification Document Production Instruction Reconciliation

An Output Specification Document Production Instruction is meaningful only in a DOCUMENT. Any Output Document Production Instruction which occurs in a BLOCK which is not a DOCUMENT shall be ignored.

If an Output Specification Document Production Instruction in Supplementary DPI, the Supplementary DPI shall take precedence over any Document DPI.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642623 218

15.8.8 Output Specification Document Production Instruction Fallback

A system which cannot perform an output specification specified by the Output Specification Document Production Instruction shall generate a structure warning. If the value of the *abort-policy* presentation parameter allows for continued processing, the effect of that continued processing shall be system dependent.

15.9 Document End Message Document Production Instruction

The Document End Message Document Production Instruction specifies a Printable String which is to be displayed on the operator interface (if any) and acknowledged after the document is printed, as close to the time of printing as possible.

15.9.1 DOCUMENT END MESSAGE DPI

The Document End Message Document Production Instruction shall be represented by a DOCUMENT END MESSAGE DPI structure element. A DOCUMENT END MESSAGE DPI structure element shall be a base structure element. The value of a DOCUMENT END MESSAGE DPI structure element shall be of type *Printable String*.

15.9.2 Document End Message Processing

The effect of a Document End Message Document Production Instruction shall be that the Printable String which is the value of the DOCUMENT END MESSAGE DPI shall be displayed on the operator interface (if any) and acknowledged after the document is printed, as close to the time of printing as possible.

15.9.3 Document End Message Reconciliation

Document End Message Document Production Instructions are meaningful only in a DOCUMENT. If a Document End Message Document Production Instruction occurs as a Supplementary DPI, the Supplementary DPI shall take precedence over the Document DPI.

15.9.4 Document End Message Fallback

Systems which cannot display a Document End Message shall indicate failure to do so by generating a structure warning.

16 Document Structure and Content Processing

Presentation of an SPDL document requires that the Presentation Process interpret the content of the SPDL document. The content of an SPDL document consists of the values of the TOKENSEQUENCE structure elements.

The processing of the content of a TOKENSEQUENCE consists of interpreting the content by the Content Processor in a *Context of Interpretation* established by the Structure Processor. Each BLOCK has an associated *Block State* which is used to generate the Context of Interpretation for the processing of the content of the BLOCK.

NOTE 64 A consequence of the properties of BLOCK boundaries is that each PAGE of an SPDL document is independent of every other PAGE.

Each PAGE also has an associated with Current Page Image. The Current Page Image is affected by the processing of TOKENSEQUENCE structure elements which are immediately subordinate to the PAGE BODY and to any subordinate PICTURE structure elements.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642624 154

16.1 BLOCK

A BLOCK structure element shall be one of.

— a PAGES structure element;

— a PICTURE structure element;

— an ENVIRONMENT RESOURCE structure element.

Every BLOCK is either a TOP LEVEL STRUCTURE structure element or subordinate to another BLOCK.

16.2 Block State

Each BLOCK has an associated *Block State.* The Block State is created at the beginning of the processing of the BLOCK and is destroyed at the completion of that processing.

The initial Block State of a BLOCK depends on whether the BLOCK is a TOP LEVEL STRUCTURE structure element or is subordinate to a superior BLOCK. The Block State of a BLOCK may be changed as a result of the processing of its subordinates.

If one BLOCK is subordinate to another BLOCK, the Block State of the subordinate BLOCK is distinct from the Block State of the superior BLOCK. Changes to the Block State of the subordinate BLOCK do not affect the Block State of the superior BLOCK.

The Block State consists of:

— a set of presentation parameters (including *abort-policy*);

— a set of defined resources;

— a set of declared resources;

— an Initial *CurrentTransformation*;

— an Inheritable Context Stack;

— a Current Virtual Machine State.

16.2.1 Current Virtual Machine State

The Current Virtual Machine State consists of:

— an Operand Stack;

— a Context Stack;

— a Graphics State Stack:

— a Current Graphics State;

— an ordered set of Save Objects;

— a set of referenced objects;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642625 090

— a set of state variables.

16.2.2 Initial Block State of a TOP LEVEL STRUCTURE

A BLOCK which is a TOP LEVEL STRUCTURE is either a DOCUMENT or an ENVIRONMENT RESOURCE. The initial Block State of a BLOCK which is a TOP LEVEL STRUCTURE shall be as follows.

16.2.2.1 Set of Presentation Parameters

The set of presentation parameters of the Initial Block State of a DOCUMENT is determined by Document Production Instructions as specified by clause 11 through clause 15. The set of presentation parameters of the Initial Block State of an ENVIRONMENT RESOURCE consists of the default values of the presentation parameters.

NOTE 65 The presentation parameter values are derived from the default values, any Supplementary DPI, and any DPI DECLARATION subordinate to the immediately subordinate PROLOGUE.

16.2.2.2 Set of Defined Resources

The set of defined resources of the Initial Block State of a TOP LEVEL STRUCTURE consists of the set of resources in the environment of the Presentation Process.

16.2.2.3 Set of Declared Resources

The set of declared resources of the Initial Block State of a TOP LEVEL STRUCTURE is empty.

NOTE 66 The default font and the single color space in the color space family *color-space-family::DeviceGray* are in the default Current Graphics State, and as such are available to content processing without being declared and without the execution of **FindFont** or **FindResource**.

16.2.2.4 Initial CurrentTransformation

The Initial *CurrentTransformation* of the Initial Block State of a TOP LEVEL STRUCTURE which is PAGESET or an ENVIRONMENT RESOURCE is specified in 19.16.

The Initial *CurrentTransformation* of the Initial Block State of a TOP LEVEL STRUCTURE which is a PAGE is specified in 16.7.3.

16.2.2.5 Inheritable Context Stack

The initial value of the Inheritable Context Stack of the Initial Block State of a TOP LEVEL STRUCTURE consists of:

<UserDictRef: DictionaryReference>

<SystemDictRef: DictionaryReference>

The Dictionaries referenced by *SystemDictRef* and *UserDictRef* are defined in 17.1.6.

16.2.2.6 Operand Stack

The Operand Stack of the Initial Block State of a TOP LEVEL STRUCTURE is empty.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

16.2.2.7 Context Stack

The Context Stack of the Initial Block State of a TOP LEVEL STRUCTURE consists of:

<UserDictRef: DictionaryReference>

<SystemDictRef: DictionaryReference>

The Dictionaries referenced by *SystemDictRef* and *UserDictRef* are defined in 17.1.6.

16.2.2.8 Graphics State Stack

The Graphics State Stack of the Initial Block State of a TOP LEVEL STRUCTURE consists of a single graphics state, which is a copy of the initial Current Graphics State.

16.2.2.9 Current Graphics State

The Current Graphics State of the Initial Block State of a TOP LEVEL STRUCTURE consists of the default values of the Graphics State Variables as specified in clause 19.

16.2.2.10 Ordered Set of SaveObjects

The Ordered Set of *SaveObjects* of the Initial Block State of a TOP LEVEL STRUCTURE is empty.

16.2.2.11 Set of Referenced Objects

The set of referenced objects of the Initial Block State of a TOP LEVEL STRUCTURE consists of the Dictionaries referenced by *SystemDict* and *UserDict* and any objects referenced directly or indirectly by the contents of those Dictionaries (see 17.1.6).

16.2.2.12 State Variables

The set of state variables of the Initial Block State of a TOP LEVEL STRUCTURE consists of the default values of the State Variables as specified in clause 19.

16.2.3 Initial Block State of a Subordinate BLOCK

The initial Block State of a BLOCK which is subordinate to a higher level BLOCK shall be as follows.

16.2.3.1 Set of Presentation Parameters

The set of presentation parameters of the Initial Block State of a subordinate BLOCK is determined by Document Production Instructions as specified by clause 11 through clause 15.

NOTE 67 With the exception of the *page-select* and *current-side* presentation parameters, the initial values of the presentation parameters are inherited from the most immediately superior BLOCK. In contrast, the initial value for each PAGESET, and the initial value of the *current-side* presentation parameter is derived from the value of the *current-side* presentation parameter for the immediately superior PAGESET as specified in 16.6.3. The processing of *page-select* presentation parameter with the *supplementary-page-select* presentation parameter are specified in clause 13.

16.2.3.2 Set of Defined Resources

The set of defined resources of the Initial Block State of a subordinate BLOCK shall be the set of defined resources in the Block State of the most immediately superior BLOCK at the beginning of the processing of the current BLOCK.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642627 963

16.2.3.3 Set of Declared Resources

The set of declared resources of the Initial Block State of a subordinate BLOCK shall be the set of declared resources in the Block State of the most immediately superior BLOCK at the beginning of the processing of the current BLOCK.

16.2.3.4 Initial CurrentTransformation

The Initial *CurrentTransformation* of the Initial Block State of a PAGESET which is subordinate to a superior BLOCK is a copy of the *CurrentTransformation* of the Current Graphics State of the Block State of the most immediately superior PAGESET at the beginning of the processing of the current PAGESET.

The Initial *CurrentTransformation* of the Initial Block State of a PAGE which is subordinate to a superior PAGESET is

$$\{ \mathrm{ICT}_p \ \mathrm{CT}_b \ \textbf{ConcatT} \}$$

where ICT_p is the initial *CurrentTransformation* determined as specified by 16.7.3 and CT_b is a copy of the Current *CurrentTransformation* of the Current Graphics State of the Block State of the most immediately superior PAGESET at the beginning of the processing of the current PAGE.

The Initial *CurrentTransformation* of the Initial Block State of a PICTURE which is subordinate to a PAGE is a copy of the *CurrentTransformation* of the Current Graphics State of the Block State of the most immediately superior PICTURE at the beginning of the processing of the current PICTURE.

16.2.3.5 Inheritable Context Stack

The initial value of the Inheritable Context Stack of the Initial Block State of a subordinate BLOCK is a copy of the Inheritable Context Stack of the Block State of the most immediately superior BLOCK at the beginning of the processing of the current BLOCK.

16.2.3.6 Operand Stack

The Operand Stack of the Initial Block State of a subordinate BLOCK is empty.

16.2.3.7 Context Stack

The Context Stack of the Initial Block State of a subordinate BLOCK is a copy of the Context Stack of the Block State of the most immediately superior BLOCK at the beginning of the processing of the current BLOCK.

16.2.3.8 Graphics State Stack

The Current Graphics State of the Initial Block State of a subordinate BLOCK is a copy of the Current Graphics State of the Block State of the most immediately superior BLOCK at the beginning of the processing of the current BLOCK.

16.2.3.9 Current Graphics State

The Current Graphics State of the Initial Block State of a subordinate BLOCK is a copy of the Current Graphics State of the Block State of the most immediately superior BLOCK at the beginning of the processing of the current BLOCK.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

16.2.3.10 Ordered Set of SaveObjects

The Ordered Set of SaveObjects of the Initial Block State of a subordinate BLOCK is empty.

16.2.3.11 Set of Referenced Objects

The Set of Referenced Objects of the Initial Block State of a subordinate BLOCK is a copy of the Set of Referenced Objects of the Block State of the most immediately superior BLOCK at the beginning of the processing of the current BLOCK.

16.2.3.12 State Variables

The values of the State Variables of the Initial Block State of a subordinate BLOCK is a copy of the values of the State Variables of the Block State of the most immediately superior BLOCK at the beginning of the processing of the current BLOCK.

16.3 Current Page Image

There is a Current Page Image created for each PAGE of the SPDL document during document processing. The Current Page Image for each PAGE is initialized to an empty page image at the beginning of PAGE processing, and is modified by the processing of TOKENSEQUENCE structure elements which are immediately subordinate to subordinate PAGE BODY structure elements or by NON-SPDL PICTURE structure elements. The processing of TOKENSEQUENCE structure elements which are not immediately subordinate to a PICTURE BODY may affect the Block State but not the state of the Current Page Image.

The Current Page Image is fully addressable during PAGE processing. A Presentation Process which can only mark on one part of the medium at a time creates a virtual page image and transfers that page image to the medium one area at a time. The actual presentation of the Current Page Image of a PAGE on the presentation surface may be done incrementally, as the TOKENSEQUENCE structure elements are processed, or the changes to the page image may be accumulated until the processing of the PAGE is complete and then transferred to the medium.

The final result of the processing of the PAGE is the rendering of the page image which is the result of the sequence of basic imaging operations on the medium. When the transfer of the page image to the presentation surface is complete, the rendered page image contains all of the effects produced by the processing of the PAGE. The effect of rendering an empty page image is the natural color of the medium on which the page image is being presented.

16.4 Interface to the Content Processor

The interface to the Content Processor is specified as an abstract procedure call. The arguments of the call to the Content Processor consist of:

— one content value;

— one Context of Interpretation;

— zero or one Current Page Image.

16.4.1 Content value

The content value represents an instance of the Content Representation and Interchange Format. The content value may be either:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642629 736

— a binary content value, consisting of a sequence of octets which represents an instance of the Binary Content Representation and Interchange Format;

— a clear text content value, consisting of sequence of octets which represents an instance of the Clear Text Content Representation and Interchange Format.

The Binary Content Representation and Interchange Format is specified in 38.9. The Clear Text Content Representation and Interchange Format is specified in 37.2.

16.4.2 Context of Interpretation

A Context of Interpretation consists of:

— a content notation identifier;

— a set of declared resources;

— an Initial *CurrentTransformation*;

— the current *abort-policy*;

— an initial Virtual Machine State.

NOTE 68 The environment of the Presentation Process and the set of defined resources are not part of the Context of Interpretation.

The Context of Interpretation used for an instance of content processing is derived from the current Block State of the most immediately superior BLOCK as described below.

16.4.2.1 Content Notation Identifier

The content notation identifier specifies whether the content value consists of a binary content value or a clear text content value.

The content notation identifier of a Context of Interpretation derived from the current Block State of a BLOCK which is an ENVIRONMENT RESOURCE identifies the content value as being

— a binary content value if the Structure Representation and Interchange Format used for the content of the BLOCK is in the Binary Structure Representation and Interchange Format;

— a clear text content value if the Structure Representation and Interchange Format used for the content of the BLOCK is the Clear Text Structure Representation and Interchange Format.

The content notation identifier of the Context of Interpretation derived from the current Block State of a BLOCK which is a PICTURE identifies the content value as being

— a binary content value if the CONTENT REPRESENTATION IDENTIFIER of the PICTURE is *content::spdl-binary*;

— a clear text content value if the CONTENT REPRESENTATION IDENTIFIER of the PICTURE is *content::spdl-clear-text*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642630 458

16.4.2.2 Set of Declared Resources

The set of declared resources is the Set of Declared Resources of the Block State of the most immediately superior BLOCK.

16.4.2.3 Initial CurrentTransformation

The Initial *CurrentTransformation* is a copy of the Initial *CurrentTransformation* of the Block State of the most immediately superior BLOCK.

16.4.2.4 Current Abort Policy

The current abort policy is a copy of the *abort-policy* presentation parameter of the set of presentation parameters of the Block State of the most immediately superior BLOCK.

16.4.2.5 Virtual Machine State

The Virtual Machine State consists of a copy of the Current Virtual machine State of the Block State of the most immediately superior BLOCK.

16.4.3 Current Page Image

A Current Page Image is created for each PAGE of the SPDL document. The Current Page Image of the most immediately superior PAGE passed to the Content Processor for the processing of the content of each TOKENSEQUENCE which is immediately subordinate to a PICTURE BODY.

16.4.4 Content Processing

Content processing consists of the interpretation of a content value as a sequence of tokens in the content notation specified by the content notation identifier, and the processing of those tokens by the Content Processor. Content processing may affect the Virtual Machine State of the Context of Interpretation which is passed to the Content Processor, and may also affect the Current Page Image if one was passed to the Content Processor as one of the arguments of the procedure call. The effect of content processing on the Virtual Machine State and the Current Page Image is specified by clause 17 through clause 36 of this International Standard.

16.4.5 Return

Upon completion of an instance of content processing, the Content Processor returns

— one Virtual Machine State;

— zero or one Current Page Image;

— status.

The returned Virtual Machine State is the result of the effects of content processing on the Virtual Machine State of the Context of Interpretation passed by the Structure Processor. The way in which the returned Virtual Machine State affects the Block State of the most immediately superior BLOCK is specified as part of the processing of the immediately superior structure elements

If a Current Page Image was passed as one of the arguments of the call to the Content Processor, it is returned at the completion of the instance of content processing. The returned Current Page Image is the result of the effects of content processing on the Current Page Image passed by the Structure Processor.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642631 394

The status shall be one of:

— *OK*;

— *content-warning*;

— *content-error*.

The meaning of *OK* is that no unhandled error or warning was generated in content processing. The meaning of *content-warning* is that one or more unhandled warnings but no unhandled errors were generated in content processing. The meaning of *content-error* is that one or more errors were generated in content processing. Exception handling is specified in clause 36.

16.5 TOKENSEQUENCE

A TOKENSEQUENCE structure element shall be a base structure element. The value of a TOKENSEQUENCE shall be a value of type *Octet String*.

Each TOKENSEQUENCE structure element is immediately subordinate to one of:

— a PICTURE BODY structure element;

— a RESOURCE SPECIFICATION structure element;

— a SETUP PROCEDURE structure element.

The processing of a TOKENSEQUENCE structure element consists of calling the Content Processor and making specified changes to the Block State of the most immediately superior BLOCK. If the TOKENSEQUENCE is immediately subordinate to a PICTURE BODY, content processing may also include making changes to the Current Page Image. The changes to the state of the most immediately superior BLOCK depend on the immediately superior structure element, and are specified as part of the processing of that structure element. A structure error or warning occurs if the status returned by the Content Processor is other than *OK*.

16.6 Association of Page Images with Instances of the Media

Medium selection and the association of page images with instances of the media is a complex process which involves several presentation parameters.

16.6.1 Page Selection and Numbering

Each PAGESET in a DOCUMENT may contain a Page Select Document Production Instruction which specifies which of the PAGE structure elements in the PAGESET are to be considered to be part of the DOCUMENT. A PAGE structure element which is excluded by a Page Select Document Production Instruction in any PAGESET is ignored for the purposes of DOCUMENT processing.

The PAGEs of the DOCUMENT are the PAGE structure elements which are not excluded by Page Select Document Production Instructions. Each of these PAGEs is assigned an *ordinal page number* in sequence which identifies the PAGE for the purposes of document presentation.

NOTE 69 These PAGEs include any PAGEs which were excluded by Supplementary Page Select DPI.

Each PAGE of the DOCUMENT which is not excluded by Supplementary Page Select DPI has associated with it a *page image* which is the Current Page Image for that PAGE at the completion of PAGE processing. Each page image is assigned an ordinal number which is the ordinal page number of the PAGE with which it was associated.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

A Supplementary Page Select Document Production Instruction in Supplementary DPI can select a subset of the PAGEs of the DOCUMENT which are not excluded by Document Page Select Document Production Instructions for a particular instance of presentation. If a Supplementary Page Select Document Production Instruction in selects a subset of the PAGEs of the DOCUMENT for a particular instance of presentation, those PAGEs presented retain the ordinal page numbers assigned to them by processing the entire DOCUMENT, including those PAGEs excluded by the Supplementary Page Select Document Production Instruction (but not those excluded by Page Select Document Production Instructions).

16.6.2 Medium Declaration and Selection

The specification of the media to be used for document presentation is a two step process:

a) Each medium which may be used for the presentation of one or more pages is first declared by a MEDIUM DECLARATION.

b) The medium which is to be used for the presentation of each page is selected by the Medium Select or Current Medium Document Production Instructions.

Medium declaration and selection are specified in clause 12. The result of medium declaration and selection is the association of a particular medium with each of the ordinal page numbers.

16.6.3 Current Side Presentation Parameter

Each PAGE of the DOCUMENT has associated with it a *current-side* presentation parameter value. The value of the *current-side* presentation parameter is either 1 or 2. The assignment of a value to the *current-side* presentation parameter for each PAGE of the DOCUMENT depends on the values of the *plex*, and *sides* presentation parameters and on the Current Side Document Production Instruction as specified below.

NOTE 70 The PAGEs to which these values are assigned include any PAGEs which were excluded by Supplementary Page Select DPI, but do not include any PAGEs which were excluded by Document Page Select DPI.

16.6.3.1 Current Side Document Production Instruction

If there is a Current Side Document Production Instruction associated with the PAGE, the value of the *current-side* presentation parameter associated with that PAGE shall be the value specified by the Current Side Document Production Instruction.

16.6.3.2 Simplex

If the value of the *plex* presentation parameter is *dpi::plex::simplex*, the values of the *current-side* presentation parameter associated with the PAGEs of the document for which there is no associated Current Side Document Production Instruction are determined as follows:

a) If the value of the *sides* presentation parameter is 1, the value the *current-side* presentation parameter associated each PAGE of the DOCUMENT shall be 1.

b) If the value of the *sides* presentation parameter is 2, the value the *current-side* presentation parameter associated each PAGE of the DOCUMENT shall be

   1) 1 if the PAGE is the first PAGE of the DOCUMENT or if the value of the *current-side* presentation parameter associated with the previous PAGE is 2;

   2) 2 if the PAGE is not the first PAGE of the DOCUMENT and if the value of the *current-side* presentation parameter associated with the previous PAGE is 1;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642633 167

16.6.3.3 Duplex or Tumble-Duplex

If the value of the *plex* presentation parameter is *dpi::plex::duplex*, or *dpi::plex::tumble*, the values of the *current-side* presentation parameter do not depend on the value of *sides*. In this case, the values of the *current-side* presentation parameter associated with the PAGEs of the document for which there is no associated Current Side Document Production Instruction shall be

a) 1 if the PAGE is the first PAGE of the DOCUMENT or if the value of the *current-side* presentation parameter associated with the previous PAGE is 2;

b) 2 if the PAGE is not the first PAGE of the DOCUMENT and if the value of the *current-side* presentation parameter associated with the previous PAGE is 1.

16.6.3.4 Other values of Plex

If the value of the *plex* presentation parameter is other than *dpi::plex::simplex*, *dpi::plex::duplex*, or *dpi::plex::tumble*, the values of the *current-side* presentation parameter associated with the PAGEs of the document shall be either:

— determined as specified by the owner of the Public Object Identifier which is the value of the *plex* presentation parameter;

— the values of *current-side* specified for *dpi::plex::simplex*.

16.6.4 Association of Page Images with Instances of the Media

Each of the page images which was associated with one of the PAGEs which was selected for the current instance of presentation as specified in 16.6.1 is presented on an instance of the medium which was associated with its ordinal page number as specified in 16.7. The page images are presented on instances of the media in their sequential order as specified below.

16.6.5 One sided printing

If the value of *sides* is 1, or if the medium or Presentation Process only supports the presentation of a single page image on each instance of the medium, each of the page images which was associated with one of the PAGEs which was selected for the current instance of presentation is presented on a separate instance of the medium.

If the value of *plex* is *dpi::plex::duplex* or *dpi::plex::tumble* and if the value of *current-side* associated with the two successive PAGEs is the same, a blank instance of the medium associated with the first of the two PAGEs shall be presented between the two PAGEs.

16.6.6 Two sided printing

If the value of *sides* is 2 and if the medium and Presentation Process support the placement of page images on two sides of each instance of the medium, the page images which were associated with each of the PAGEs which were selected for the current instance of presentation are presented alternately on the first (obverse) side and the second (reverse) side of instances of the media. Page images are associated with the presentation surfaces of the instances of the media sequentially in the order specified by the ordinal page numbers. The medium on which each page image is presented is the medium which was associated with the ordinal page number which was associated with the page image.

The Current Page Image associated with a PAGE is presented on a new instance of the medium if any of the following is true:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642634 0T3

— the value of the *current-side* presentation parameter associated with the PAGE is 1;

— the value of the *current-side* presentation parameter associated with the previous PAGE was 2;

— the ordinal page number of the current PAGE is other than the ordinal page number of the previous PAGE plus one;

— the medium associated with the ordinal page number of the current PAGE is different from the medium on which the page image for the previous PAGE was placed.

The last condition is satisfied only when the actual media on which the two page images is placed are placed are different. If the same medium is assigned to two different MEDIUM SPECIFICATIONs, consecutive page images shall not be placed on different instances of the medium simply because the MEDIUM SPECIFICATIONs are different

If the value of the *current-side* presentation parameter associated with the PAGE is 1, the page image is presented on the first (obverse) side of the medium. If the value of the *current-side* presentation parameter associated with the PAGE is 2, the page image is presented on the second (reverse) side of the medium. Further:

— if a page image is presented on a new instance of the medium and the previous page image was placed on the obverse side of another instance of the medium, the reverse side of that instance of the medium is left blank;

— if a page image is presented on the reverse side of a new instance of the medium, the obverse side of that instance of the medium is left blank.

16.7 Placement of the Page Image on the Medium

The placement of the page image generated by a PAGE on an imageable surface of the medium is determined by

— the association of the page image with an imageable surface of an instance of the medium as described in 16.6;

— the location and orientation of the Reference Coordinate System of the page image on the medium as specified in this clause.

16.7.1 Location and Orientation of the Reference Coordinate System on the Medium

Each imageable surface has associated with it

— a reference orientation;

— a medium Reference Coordinate System;

— two values, *MediumXSize* and *MediumYSize*, which indicate the size of the medium.

The reference orientation and the location and orientation of the medium Reference Coordinate System on the medium are functions of the medium and not of the page image.

NOTE 71 The location and orientation of the medium Reference Coordinate System on the medium is the same even if the page images are being placed on the medium in such a way that the completed document will commonly be viewed in an orientation other than the reference orientation (e g. with the long edge horizontal in landscape orientation).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

16.7.1.1 Reference Orientation

For media which have a distinguished natural viewing orientation, the reference orientation shall be the natural viewing orientation of the medium.

For media such as paper which have no distinguished natural viewing orientation, the reference orientation shall be with the long edge vertical (portrait orientation).

16.7.1.2 Medium Reference Coordinate System for Rectangular Media

For rectangular media, the location and orientation of the medium Reference Coordinate System on the medium shall be such that with the medium oriented in its reference orientation:

— the origin of the Reference Coordinate System is at the bottom left corner of the medium;

— the *x*-axis of the Reference Coordinate System is placed along the bottom edge with positive values of *x* to the right;

— the *y*-axis of the Reference Coordinate System is placed along the left edge of the medium with positive values of *y* in the upward direction.

16.7.1.3 Medium Reference Coordinate System for Non-Rectangular Media

For non-rectangular media, the location and orientation of the medium Reference Coordinate System on the medium is a property of the medium, but should approximate the placement of the Reference Coordinate System for rectangular media

16.7.1.4 Two Sided Media

When page images are placed on both sides of a two sided medium, the reference orientation for the second (reverse) side shall be the orientation that results from rotating the medium around an axis which coincides with the *y*-axis of the medium Reference Coordinate System for the first (obverse) side.

16.7.1.5 MediumXSize and MediumYSize

The values of *MediumXSize* and *MediumYSize* associated with each medium represent the extent of the medium in the medium Reference Coordinate System. The values of *MediumXSize* and *MediumYSize* (in the units of the current User Coordinate System) are available to content processing as values of the *CurrentXMedium* and *CurrentYMedium* entries in the Device Description Dictionary (see 19.8). The values of *MediumXSize* and *MediumYSize* associated with each medium shall be such that the imageable region of the medium is contained within the rectangle bounded by the medium Reference Coordinate System and the values of *MediumXSize* and *MediumYSize*.

16.7.1.6 Virtual Presentation Surfaces

Softcopy devices may place page images on virtual presentation surfaces which can then be viewed on the physical presentation surface. For such devices, each virtual presentation surface is treated as a separate instance of the medium. The mapping of the virtual presentation surfaces to the physical display surface and the ability of the system to preserve a virtual presentation surface for future viewing are system dependent.

16.7.2 Placement of the Page Image on the Imageable Surface of the Medium

The placement of the Current Page Image for the PAGE on the imageable surface of the medium depends on the value of the *plex* presentation parameter.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642636 976

16.7.2.1 Simplex or Duplex

If the value of the *plex* presentation parameter is *dpi::plex::simplex* or *dpi::plex::duplex*, the Current Page Image is placed on the imageable surface of the medium so that the Reference Coordinate System of the Current Page Image coincides with the medium Reference Coordinate System.

16.7.2.2 Tumble Duplex

If the value of the *plex* presentation parameter is *dpi::plex::tumble*, the page image associated a PAGE for which *current-side* is 1 is placed on the imageable surface of the medium so that the Reference Coordinate System of the Current Page Image coincides with the medium Reference Coordinate System. The page image associated a PAGE for which *current-side* is 2 is placed on the imageable surface of the medium so that:

— the origin of the Reference Coordinate System of the Current Page Image coincides with the point whose coordinates are (*MediumXSize, MediumYSize)* in the medium Reference Coordinate System;

— the point whose coordinates are (0, *MediumYSize*) in the Reference Coordinate System of the Current Page Image coincides with the point whose coordinates are (*MediumXSize*, 0) in the medium Reference Coordinate System;

— the point whose coordinates are (*MediumXSize*, 0) in the Reference Coordinate System of the Current Page Image coincides with the point whose coordinates are (0, *MediumYSize)* in the medium Reference Coordinate System.

NOTES

72 The effect is that the page images placed on the reverse side of the medium are rotated 180 degrees.

73 The specified page image rotation is used in one sided printing as well as two sided printing. This enables the output from one sided printing with *plex=dpi plex::tumble* to be copied in tumble duplex form on a 1-side-to-2-side copier with long-edge feed.

16.7.2.3 Other values of plex

If the value of the *plex* presentation parameter is other than *dpi::plex::simplex*, *dpi::plex::duplex*, or *dpi::plex.:tumble*, the placement of the Current Page Image on the imageable surface of the medium shall be either:

— the placement specified by the owner of the Public Object Identifier which is the value of the *plex* presentation parameter (if known);

— the placement specified for *plex=dpi:.plex' simplex*.

16.7.3 Initial CurrentTransformation

The *CurrentTransformation* of the initial Current Graphics State and the Initial Graphics State of the PAGE shall be referred to below as the *Initial CurrentTransformation*. The Initial *CurrentTransformation* depends on the values of the *plex*, *current-side*, *x-image-shift*, and *y-image-shift* presentation parameters. If for a particular PAGE:

— the value of the *x-image-shift* presentation parameter is *x*;

— the value of the *y-image-shift* presentation parameter is *y*;

— the value of the *current-side* presentation parameter is *side*;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642637 802

then Initial *CurrentTransformation* depends on the value of the *plex* presentation parameter and these values as follows.

16.7.3.1 Simplex

If the value of the *plex* presentation parameter is *dpi::plex::simplex*, the Initial *CurrentTransformation* for each PAGE shall be

{*x y* **TranslateT**}.

16.7.3.2 Duplex

If the value of the *plex* presentation parameter is *dpi::plex::duplex*, the Initial *CurrentTransformation* for each PAGE shall be

{*x side* 2 **Equal** {**Negate**} **If** *y* **TranslateT**}

16.7.3.3 Tumble Duplex

If the value of the *plex* presentation parameter is *dpi::plex::tumble*, the Initial *CurrentTransformation* for each PAGE shall be

{*x y side* 2 **Equal** {**Negate**} **If TranslateT**}

16.7.3.4 Other values of Plex

If the value of the *plex* presentation parameter is other than *dpi::plex::simplex*, *dpi::plex::duplex*, or *dpi::plex::tumble*, the Initial *CurrentTransformation* for each PAGE shall be either:

— the initial *CurrentTransformation* specified by the owner of the Public Object Identifier which is the value of the *plex* presentation parameter (if known);

— the initial *CurrentTransformation* specified for *plex=dpi::plex::simplex.*

17 Document Content Processing Model

Content processing consists of parsing the Content value and processing the resulting sequence of tokens. Unlike the structure notation, which is declarative in nature, the Content values express the document contents in the form of a procedural language to be interpreted by the Content Processor. The effect of processing a Content value is described in terms of a Virtual Machine. The Content values, and their associated Contexts of Interpretation, are identified during Structure Processing as described in clause 16 . These data are then provided to the Content Processor.

The Content values identified by the Structure Processing are parsed, by the Virtual Machine, into a sequence of tokens as specified in 38.2 if the Content value is in Binary Content Representation and Interchange Format and in 37.2 if the Content value is in Clear Text Content Representation and Interchange Format as indicated by the content notation specifier in the Content value.

Processing the sequence of tokens consists of interpreting each token and modifying the Virtual Machine State and the Current Page Image as specified for that token. Some tokens have no effect on the Current Page Image and others may make large modifications. Defining this complex activity more precisely requires several sub models.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

This Clause defines the Virtual Machine, and the imaging model, and coordinate systems. The Virtual Machine models the overall content interpretation process. The imaging model defines how the Current Page Image is modified and the coordinate system provides the frame of reference for operators that affect the Current Page Image.

This International Standard does not specify any fallback processing to be invoked when the presentation device does not have the capabilities expected of that device when the Document was composed. Such actions can be included in SPDL Documents by using content operators to test the characteristics of the device and conditionally execute content Procedures based on the results of those tests.

17.1 Virtual Machine

The Virtual Machine models the overall process of content processing.

NOTE 74 The Virtual Machine model is presented only as a model for defining the semantics of document content. It is not intended to dictate the actual implementation of a document content processor, nor to restrict the method by which document content semantics are realized in a particular implementation.

The Virtual Machine model of a document content processor consists of the following components:

— parser

— interpreter

— sequence of tokens

— Operand Stack

— Context Stack

— set of Referenced Objects

— State Variables

— Graphics State and Graphics State Stack

— Ordered set of SaveObjects

— imager

— Current Page Image

— abort policy

— set of Declared Resources

— initial current transformation

Figure 9 shows the components of the Virtual Machine and states the subclauses in which they are specified.

The remainder of this subclause is a more detailed definition of the components of the Virtual Machine. First, some concepts used in the definitions of the individual components are introduced and then the definitions of the components themselves follow.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

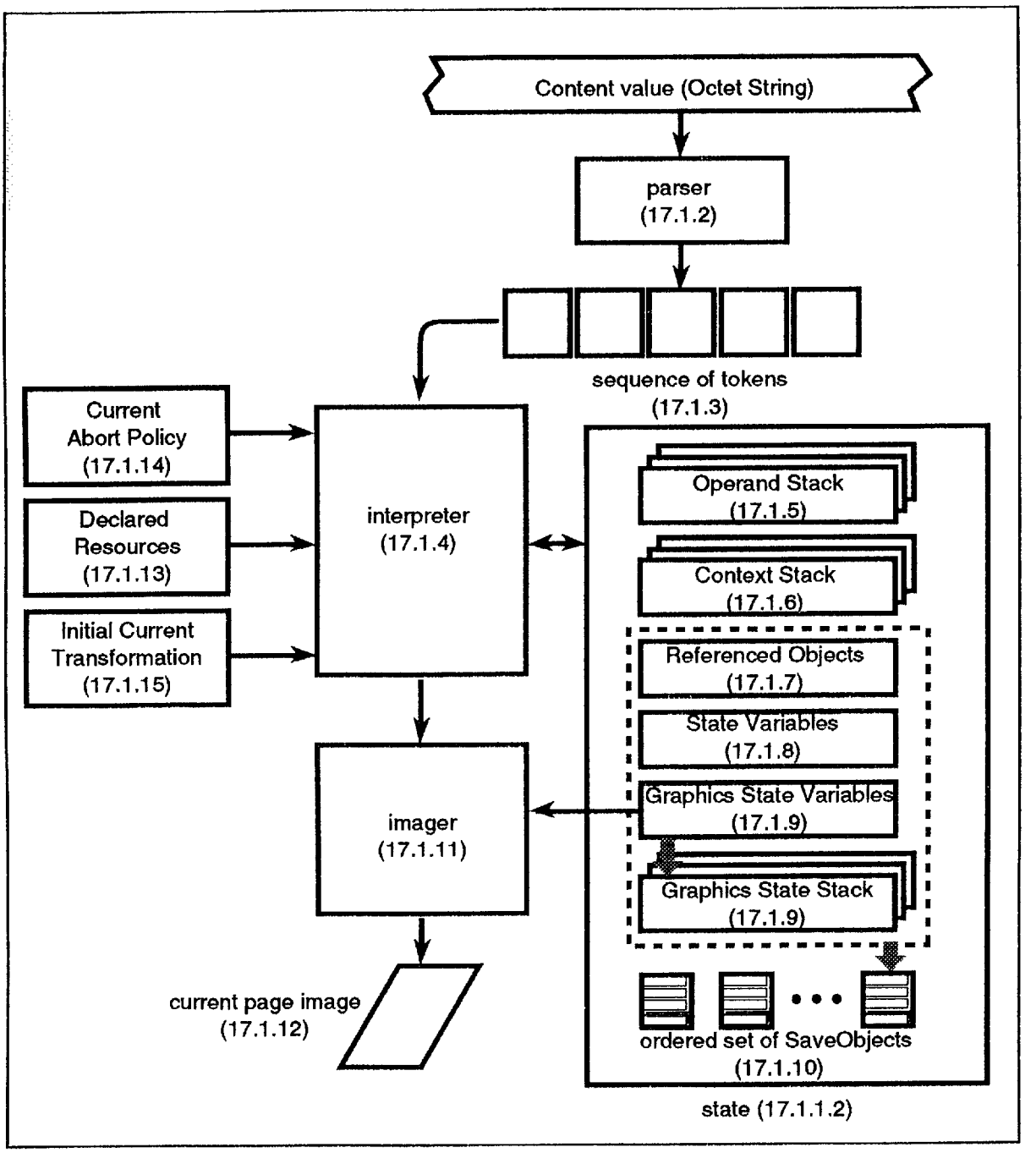

Figure 9 — Virtual Machine

17.1.1 Basic Concepts

The definitions of the components of the Virtual Machine are based on three concepts: object, state, and operator.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642640 3T7

17.1.1.1 Objects

The data elements that can be operated upon by the Virtual Machine are objects. The class of *objects* is an expansion of the class of values defined in clause 6 . The class of objects shall be partitioned into two subclasses: values and composite objects. The class of values is extended with a new set of objects called *object references*. Where it is necessary to identify the values that are not object references, these will be called *simple values*. Only values, simple and object references, can be used directly as operands of content operators.

Composite objects possess properties that values do not have. A composite object has substructure; it is a multipart container which can hold values (including object references). The set of values held is called the *contents* of the composite object. Object references are references to other such containers. Composite objects are created by some operator or action and are identified by the object reference that results from the creation.

The semantics of a value are the same wherever it occurs. The semantics of the integer 2 do not depend on how that 2 was generated. The semantics of a composite object, however, depend on its identity as well as its content. If two object references identify the same composite object, then they are equal; otherwise they are not equal even if the content of the referenced composite objects is the same. An object reference that identifies a composite object will be said to *refer* to that composite object.

A composite object has visible substructure if there is some content operator that will modify a part of the composite object without generating a new object. This means that if there are two object references that refer to the same composite object, then using one as the operand of such a modifying operator will produce a change in the composite object that is visible when the other object reference is used to access that composite object The two object references will be said to share the composite object.

Composite objects are also distinguished from values by their behavior when copied. Copying a value yields another instance of the value type which is indistinguishable from the original instance. In fact, one cannot tell whether there really is a second instance at all. If the value was an object reference, then the two object references share the same composite object.

Copying a composite object, however, requires using another container with its own identity (or creating a new container) and copying the content of the original object into the second container. These are two distinct composite objects so the references to them do not share.

Each object has a *type*. The types of values include some of the types defined in clause 6 plus some new types. New types are introduced for the class of Composite objects. For each composite object type *X*, there is a corresponding *XReference* object reference type. The object types defined by this International Standard are specified in clause 18

17.1.1.2 State

The Virtual Machine *State* consists of the Operand Stack, the Context Stack, the Graphics State Variables, the Graphics State Stack, the set of State Variables, the ordered set of SaveObjects, and the set of Referenced Objects. The Virtual Machine State consists of all but one of the components of the Virtual Machine that can be changed by content processing; the remaining component that can be changed by content processing is the Current Page Image.

The Virtual Machine State:

— is initialized from the Context of Interpretation provided by the Structure Processor when the Virtual Machine is called,

— can be changed by operators performed by the Virtual Machine, and

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642641 233

— is used to construct the resulting Context of Interpretation when the processing of the content is completed.

17.1.1.3 Operators

Operators cause actions to be taken by the Virtual Machine. These include:

— manipulating objects

— manipulating the Virtual Machine State

— imaging activities

Operators have zero or more operands which are:

— either explicit as values on the Operand Stack;

— or implicit as some parts of the rest of the Virtual Machine State prior to the execution of the operator.

The action that an operator takes never depends on the state of the Current Page Image prior to interpreting the operator.

17.1.2 Parser

The procedural input to content processing is the Content value which corresponds to a TOKENSEQUENCE that was identified by the structure processor. This Content value contains an Octet String with data in one of the two Content Representation and Interchange Formats defined in clause 37 and clause 38. The *parser* converts the Octet String into a sequence of tokens using the rules defined in the appropriate Content Interchange Format definition (37.2 or 38.2) as determined by the content notation specifier in the Content value.

NOTE 75 *the parser can be executed in parallel with the interpreter (see 17.1.4 below).*

17.1.3 Sequence of tokens

The input to the interpreter is this sequence of *tokens*. Each token is the object that results from the parser processing the token encoding. If a value was encoded, the result is a value; if a composite object was encoded, the result is an object reference to the composite object. In either case, the result is a value.

Some values may have an *executability attribute* as defined in 18.4.1 . The executability attribute may be either *literal* or *executable* and the value is said to be a *literal value* or an *executable value*, respectively. For those values with an executability attribute, the parser, using the rules specified in clause 37 and clause 38, determines the executability attribute.

In addition to interpreting content token sequences, it is possible to save a subsequence of the tokens as a Procedure. A Procedure is an executable object reference to a Vector object (see 18.2.4), the elements of which are the values produced by parsing the saved tokens. Execution of a Procedure object provides an alternate source of objects to the interpreter.

17.1.4 Interpreter

The interpreter takes as its input a sequence of values. This sequence may either be a token sequence or the sequence of elements of an executable Vector (a Procedure). The interpreter interprets each such value in the order of the given sequence.

Interpreting a literal Identifier, a literal VectorReference, or any other type of value that does not possess an executability attribute causes that value to be pushed onto the Operand Stack.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642642 17T

Interpreting an executable Identifier object causes the value bound to that identifier to be interpreted. Finding the value bound to an identifier is described in the Context Stack description, below.

The interpretation of an executable Vector reference (a Procedure reference) depends on how that reference is encountered. If the reference is encountered *directly* in the token sequence or as an element of an executable Vector, then the encountered executable Vector reference is pushed onto the Operand Stack and is not executed. If the executable Vector reference is encountered *indirectly* as the result of the execution of some Identifier or Operator, then the executable Vector that is referenced is executed.

Interpreting an object of type *Operator* causes the Virtual Machine to perform actions that depend on the particular operator. The set of operators defined by this International Standard is defined in Clauses clause 20 to clause 36.

NOTE 76 The set of operators is summarized in annex A.

The actions resulting from interpreting an operator may include:

— removing values (to be used as operands) from the Operand Stack

— performing calculations or other manipulations upon objects

— creating new objects

— placing values (as results) on the Operand Stack

— querying or changing Dictionaries

— querying or changing the Virtual Machine State

— referencing and using resources in the Context of Interpretation

— causing imaging activities

The actions specific to each operator are defined along with the operators in clause 20 to clause 36.

17.1.5 Operand Stack

The Operand Stack is a stack, which may hold any number of values. It is a heterogeneous stack, in that it may hold any of the types of values in any combination. The Operand Stack is initialized from the Operand Stack in the Virtual Machine State in the Context of Interpretation passed to the Content Processor.

NOTE 77 In this International Standard, the terms *stack*, *top of stack*, *push*, and *pop* are used in a way consistent with common computer science usage.

Operators requiring values as explicit operands pop them from the operand stack, and operators which return values as explicit results push them on the Operand Stack.

17.1.6 Context Stack

The Context Stack is a stack of Dictionaries and is a component of the Virtual Machine State. Dictionaries (defined in 18.2.1) are lists of key/value pairs in which the key is an Identifier or Cardinal and the value can be a simple value or object reference.

The Context Stack is an ordered list of Dictionaries. The initial Context Stack is taken from the Virtual Machine State in the Context of Interpretation passed to the Content Processor . A dictionary can be added to the top of the Context Stack with the **PushContextStack** operator. Dictionaries can be removed from the Context Stack

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642643 006

with the **PopContextStack** operator. Neither *UserDict* nor *SystemDict* can be removed from the Context Stack.

The Context Stack is used to find the value bound to an Identifier. This is also known as the *binding* for that Identifier. When the context stack is referenced to find the binding for an Identifier, it is searched in reverse order: the last Dictionary is searched first; if the Identifier is a key in that Dictionary, then the value paired with that key is returned; otherwise, the next Dictionary is searched, and so on. If no Dictionary on the Context Stack has the Identifier as a key, then **RaiseError** is invoked with *UndefinedKey* as its operand.

The interpretation of executable Identifiers implicitly uses the Context Stack to retrieve the object bound to the Identifier. The **GetValue** operator can be used with a literal Identifier or Cardinal to explicitly retrieve the object bound to that key.

17.1.6.1 Standard Dictionaries

There are two standard dictionaries which occur on the Context Stack:

— *SystemDict* This Dictionary shall contain a key/value pair for each SPDL operator. For each key/value pair, the key is the name of the operator as defined in annex A of this International Standard, and the value associated with key is an object of type *Operator* which implements the functions of the operator. The Dictionary shall also contain four other key/value pairs. First, a key/value pair whose key is the name *SystemDict*, and whose value is of type *DictionaryReference* which references *SystemDict*. Second, a key/value pair whose key is the name *UserDict*, and whose value, of type *DictionaryReference*, references the context Dictionary *UserDict* (see below). Third a key/value pair whose key is the name *ErrorDict*, and whose value, of type *DictionaryReference*, references the Dictionary *ErrorDict*. *ErrorDict* is a modifiable Dictionary defined in clause 36. Fourth, a key/value pair whose key is the name *ErrorInfoDict*, and whose value, of type *DictionaryReference*, references the Dictionary *ErrorInfoDict*. *ErrorInfoDict* is a modifiable Dictionary defined in clause 36. *SystemDict* is always the first Dictionary on the Context Stack and cannot be modified by an SPDL document.

— *UserDict* This Dictionary has no predefined key/value pairs, but may be modified by content processing and shall be capable of holding the maximum number of name/value pairs that can be held by any dictionary. *UserDict* is always the second Dictionary on the default Context Stack.

17.1.7 Set of Referenced Objects

The set of Referenced Objects is a component of the Virtual Machine State. This set consists of the set of all composite objects that were present in the Context of Interpretation or have been created by interpretation of the content, and are still referenced. A composite object is *referenced* if an object reference to that object is the value paired with some key in a Dictionary on the Context Stack, is on the Operand Stack, is the value of a Graphics State Variable, is the value of a Graphics State Variable saved on the Graphics State Stack, is the value of a State Variable, or if an object reference to it is a value stored in some other referenced object. The initial set of Referenced Objects is taken from the Virtual Machine State in the Context of Interpretation passed to the Content Processor.

17.1.8 State Variables

A set of *State Variables* is a component of the Virtual Machine. The State Variables are *global* in that they can always be referenced from content. The values of the State Variables form part of the Virtual Machine State. The State Variables and their initial values are defined in clause 19.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642644 T42

17.1.9 Graphics State and the Graphics State Stack

In addition to the State Variables there is a separate set of global variables representing the state of the imager. These are the *Graphics State Variables*. They act as implicit arguments to most of the imaging operators. The Graphics State Variables and their initial values are defined in clause 19. Examples include *CurrentTransformation* and *CurrentColor*. The complete set of Graphics State Variables and their values is known as the *graphics state*.

The Graphics State Stack is a component of the Virtual Machine State. The set of Graphics State Variables can be saved on this stack by the **SaveGraphicsState** operator. This saved graphics state can be restored either explicitly by the **RestoreGraphicsState** or **RestoreGraphicsStateXCP** operator, or if it is the graphics state that is part of a SaveObject, explicitly by **RestoreSavedGraphicsState** and implicitly by the **RestoreState** operator. Unlike the Operand Stack, there are no other operators which explicitly manipulate this stack. The objects on the stack are implementation-specific data objects which represent the values of the Graphics State Variables at the time the object on the stack was created. The Graphics State Stack is initialized from the Graphics State Stack in the Virtual Machine State in the Context of Interpretation passed to the Content Processor. The operators that affect this stack are defined in clause 22 .

17.1.10 Ordered set of SaveObjects

The values of the referenced objects, the State Variables, the Graphics State Variables and the Graphics State Stack can be explicitly saved using the operator, **SaveState**. This operator returns a SaveObject which can later be restored, using **RestoreState**. All composite objects, including any subsequent SaveObjects, created after the interpretation of a **SaveState** operator will be destroyed when its resulting SaveObject is restored. All other composite objects will be restored to the status they had at the time the **SaveState** operator is interpreted.

NOTES

78 The operator **SaveState** does not save the state of the Context Stack nor the Operand Stack. The Context Stack may be saved by transferring a copy of its contents to a Vector using the operator **ContextStack**. The Operand Stack can be saved in a Vector using a simple Procedure that uses the operators **Count**, **Mark**, **Copy**, **Pop**, and **MakeandStoreVector**.

79 The management of storage usage in an SPDL Interpreter implementation is largely left to the designer of the implementation. The operators that create composite objects and the destruction and restoration of objects during Restore establish the constraints that any storage management system shall operate within.

80 A SaveObject must be retained either on the operand stack or as a value in some referenced object if it is to be used later as an operand for **RestoreState**. This implies that the implementation of that SaveObject (the storage used) must be saved as part of the interpretation of any subsequent **SaveState** operator.

There is no explicit formal structure to the set of SaveObjects. However, the action of **RestoreState** means that there is, effectively, an ordering defined on the set of SaveObjects making them an *ordered set of SaveObjects*. This ordering begins with the oldest unrestored SaveObject and ends with the most recent unrestored SaveObject. Restoring a SaveObject removes that *SaveObject* and all the SaveObjects subsequent to that SaveObject in the ordering. The initial value of the ordered set of *SaveObjects* is taken from the Virtual Machine State in the Context of Interpretation. There is always at least one SaveObject in the ordered set of SaveObjects: the implicit *SaveObject* for the most immediately superior BLOCK.

17.1.11 Imager

The *imager* is the part of the interpreter that interprets the imaging operators and causes imaging activities on the Current Page Image. The imager uses the values of the Graphics State Variables as implicit operands of the imaging operators. The operators for imaging text, geometric graphics, and raster graphics are defined in clause 27, clause 28 and clause 29; additional imaging operators are defined in clause 32 and clause 33.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

17.1.12 Current Page Image

The *Current Page Image* is provided as an argument to Content Processor and may be absent. The Current Page Image is a page image. If present, the Current Page Image is changed by the interpretation of imaging operators as described below in 17.2.4 and the changed page image is returned as a result of content processing. If no Current Page Image was provided, any imaging operators that normally perform an imaging activity when interpreted do not do so (and give no error or warning that no imaging activity was performed).

At the beginning of processing of the content of a PAGE, the Current Page Image starts out *empty*. It is the responsibility of the document Content Processor to build up the Current Page Image as it interprets imaging operators. It is the responsibility of the Structure Processor to associate complete page images with instances of the presentation media as described in clause 16.

17.1.13 Set of Declared Resources

The set of Declared Resources is part of the Context of Interpretation, which is constructed during document structure processing and passed when the Content Processor is invoked. The *set of Declared Resources* is the set of all resources that have been declared to be available while processing the Content value supplied to this invocation of the Content Processor. The set of Declared Resources cannot be changed during content processing.

A resource is referenced from within content by using *resource-finding* operators, either the generic **FindResource** operator or an operator specific to the given resource type. Such operators take as an operand the Name of the resource, which is the INTERNAL RESOURCE NAME bound to the resource in the RESOURCE DECLARATION. Such operators return the corresponding resource as an object on the Operand Stack. The *methods for defining resources at the structure level are defined in clause 9 clause 10 . The semantics of the* values of the various types of resources are defined in clause 24 through clause 27, and clause 31 through clause 34. The **FindResource** operator is defined in clause 21.

17.1.14 Current Abort Policy

The Current Abort Policy is taken from the *abort-policy* presentation parameter in the *Block State* of the most immediately superior BLOCK when the Context of Interpretation is constructed. The Current Abort Policy is provided in the Context of Interpretation to control the action of exception handling within Content. It is used to determine whether execution continues after a Warning occurs or whether processing of the whole document is aborted at that point. If the Current Abort Policy is *dpi::abort-policy::on-warning*, further processing of the Document and the Content value is aborted. Otherwise, processing continues after a Warning. The Current Abort Policy cannot be changed by content processing.

17.1.15 Initial *CurrentTransformation*

During content processing, the coordinate system currently in effect can be changed by content operators, see 17.3 below and clause 23 . The range of these changes shall be limited by the Initial *CurrentTransformation* which is provided in the Context of Interpretation. The Initial *CurrentTransformation* cannot be changed by content processing.

17.2 Imaging Model

The interpretation of some of the imaging operators changes the Current Page Image. Such operators perform imaging activities. Imaging activities make changes to the page image in accordance with the imaging model.

The *imaging model* is a two dimensional graphics model appropriate for device independent usage, and for high quality printing and publishing needs. A page image is built incrementally by a sequence of imaging activities beginning with an initially empty page image. Each imaging activity places an *image element* on the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642646 815

page image. An *imaging activity* places an image element on the page image by painting ink through a mask and a clipping region, as shown in figure 10. The Current Page Image is the page image, if any, that is a component of the Virtual Machine (see 17.1.12).

There are four components to the imaging model:

— ink

— mask

— clipping region

— page image

Every imaging operator that adds an image element to a page image involves an instance of each of these components: the mask is constructed by or provided to the imaging operator, the ink and the clipping region are taken from the Graphics State Variables, and the page image is the Current Page Image component of the Virtual Machine. These components are described further below.

17.2.1 Ink

An imaging activity causes the currently selected color in the currently selected color space to be painted onto the page image as an *ink*.

The term *ink* refers to the color and visual texture which will be displayed by the presentation device. This International Standard defines two kinds of inks: solid inks and patterns

A *solid ink* has a single color which can be any of the colors expressible in the *CurrentColorSpace*, including black, white, or any shade of gray. Solid inks are logically opaque, and obscure the colors of previously placed image elements occupying the same area of the page image. Color spaces are defined in clause 34 .

A *pattern* need not be solid but instead consists of repeated instances of a graphic figure, the pattern cell. The definition of the pattern specifies the coordinate space in which the pattern is defined, the X and Y distances between repeated instances and a Procedure that draws the pattern cell. The pattern cell may have transparent areas. In addition, the pattern cell may itself be already colored or it may simply be a mask to which a solid color must be associated before painting with that pattern. Clause 32 has the definition of patterns.

With the exception of the **ImageRasterElement** operator, the color to be painted for each imaging operator, and the color space to which it refers, are taken from the *CurrentColor* and *CurrentColorSpace* Graphics State Variables, respectively. In the case of the **ImageRasterElement** operator, the color space is taken from *CurrentColorSpace* which must not be a pattern color space. The color to be painted for each individual sample in the sampled raster graphics image that is the operand of the **ImageRasterElement** operator is specified by the sample value which specifies a color in the *CurrentColorSpace*; see clause 28.

17.2.2 Mask

The *mask* provides the shape of the image element. A mask may be described as the interior of a geometric path, as the area stroked by a pen following a geometric path, an instance of sampled raster graphics image data, or an instance of bitmap raster graphics image data. In addition, each reference to a glyph in a font is an indirect reference to the mask determined by the glyph representation. Geometric paths may be used to describe a wide variety of shapes, including simple convex shapes, concave shapes, or shapes with self-intersecting or disjoint boundaries All of these types of mask may be described by imaging operators.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

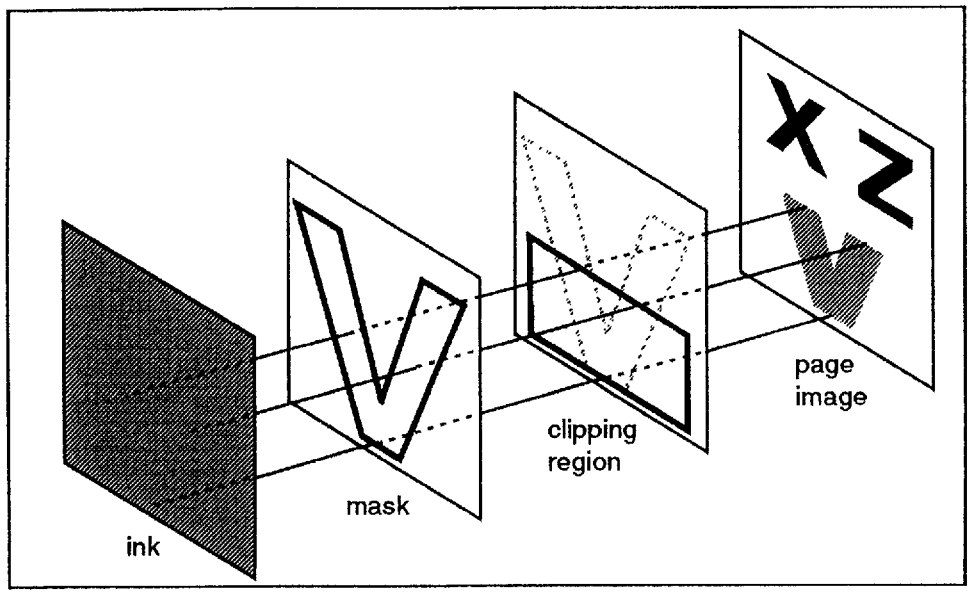

Figure 10 — Imaging Model

The mask determines where ink is placed on a page image. Without a clipping region, ink would change the page image where and only where the mask is defined. The clipping region further limits where the page image is changed.

17.2.3 Clipping Region

The clipping region is the region of a page image upon which ink may be placed. Ink falling within the clipping region will affect the page image, while ink falling outside the clipping region will not. Thus the shape of the image element may be *clipped* by the clipping region.

The boundary of the clipping region is described using the same facilities used to describe masks geometrically. Initially, the clipping region is a rectangle containing the entire imageable region of the page. However, it may be further restricted by intersection with another geometric region. Changes to the clipping region may only be undone by restoring a Graphics State which was saved before the clipping region was changed.

The clipping region used for each imaging operator is taken from the *CurrentClippingRegion* Graphics State Variable.

NOTE 81 The *assured reproduction area* of an page image is that region of the page image upon which marks may be made. The position and size of this region with respect to the Reference Coordinate System will vary with different media and different SPDL implementations; however, SPDL provides State Variables that allow determination of its position and size (see 19.19) during interpretation of content. The assured reproduction area is the default *CurrentClippingRegion* for a page image that is associated with a given medium, and is always within the extents of that medium.

17.2.4 Page Image

The *page image* is an abstract, device independent model of the page that is to be rendered on a medium. A page image is built incrementally by sequentially adding image elements to an initially empty page image. An *image element* is the result of a single imaging activity. Imaging activities defined in this International Standard include painting

— a filled geometric region,

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642648 698

— a stroked geometric region,

— an instance of a sampled or bitmap raster graphic,

— a single glyph

through the *CurrentClippingRegion* onto the page image. The construction of an image element and any other effects of imaging operators do not in any way depend on the content or the state of that page image at the time the imaging operator is interpreted.

17.3 Coordinate Systems

The content imaging operators are all interpreted in the context of a current User Coordinate System. This coordinate system is defined in terms of a Reference Coordinate System established by the Structure Processor in 16.7.1 . The relationship of the current User Coordinate System to the Reference Coordinate System is established by the *CurrentTransformation* Graphics State Variable. The Initial *CurrentTransformation* for each PAGE is established by the Structure Processor in 16.7.3 . In addition, the *CurrentTransformation* in effect when each subordinate PICTURE on the page is processed is saved as the Initial *CurrentTransformation* for that BLOCK and is passed as the Initial *CurrentTransformation* in conjunction with the content processing for all TOKENSEQUENCES immediately subordinate to that BLOCK. This Initial *CurrentTransformation* establishes a coordinate system in terms of which all coordinate transformation within that BLOCK are relative.

Locations within the Reference Coordinate System or the User Coordinate System are given either by separately specifying the X and Y coordinates of the location, or in terms of a point. A *point* is a pair of values in which the first value is the X coordinate of the location and the second value is the Y coordinate of the location.

17.3.1 Reference Coordinate System

The Reference Coordinate System (*RCS*) provides for unambiguous specification of locations on a page image.

The RCS is a two dimensional Cartesian coordinate system with units of millimeters on both axes. The positioning of the RCS on the Current Page Image is specified in 16.8.

17.3.2 User Coordinate Systems

A User Coordinate System (*UCS*) provides for specification of locations on the Current Page Image in a flexible way.

All position information used by imaging operators is initially specified in terms of a UCS, and a mapping from the UCS to the RCS. The mapping performs two functions:

— it associates physical measurement units of the RCS with the dimensionless units of the UCS;

— it provides the user with the freedom to describe locations and shapes with respect to a conveniently chosen UCS. The UCS used for such descriptions will not constrain the size or orientation of the resulting shapes when these descriptions are used to add image elements to the Current Page Image, because the mapping can be changed arbitrarily.

This mapping is provided by the *CurrentTransformation* Graphics State Variable. The *CurrentTransformation* can specify any mapping of the UCS into the RCS that can be represented in the form:

$$x' = \mathrm{a} \times x + \mathrm{c} \times y + \mathrm{e}$$

$$y' = \mathrm{b} \times x + \mathrm{d} \times y + \mathrm{f}$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

where a, b, c, d, e, and f are real number constants whose dimensions are millimeters, x and y are coordinates in the UCS, and x′ and y′ are coordinates in the RCS. This is equivalent to the matrix equation

$$[\,x'\ y'\ 1\,] = [\,x\ y\ 1\,] \times \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix}$$

NOTE 82 Any such mapping where $a \times d - b \times c = 0$ will be degenerate and non-invertible.

The set of all transformations expressible in terms of the six numbers a, b, c, d, e, and f is the set of *affine transformations* and includes all combinations of scaling, translation and rotation. The effect of the *CurrentTransformation* is to map an image element defined in *UCS Space* to an image element that is defined in *RCS Space* and can be added to the Current Page Image.

The operators which create transformation objects (suitable for assigning to the *CurrentTransformation* Graphics State Variable) are defined in clause 23.

17.3.3 Initial *CurrentTransformation*

At any instant in time during content processing, the *CurrentTransformation* is logically composed of two parts: the Initial *CurrentTransformation* for the most immediately superior BLOCK to the content being processed and the Active *CurrentTransformation*. The Initial *CurrentTransformation* is established by the Context of Interpretation. The Active *CurrentTransformation* is the product (by matrix multiplication) of the *CurrentTransformation* times the matrix inverse of the Initial *CurrentTransformation*. Content processing can only change the Active *CurrentTransformation*; it can never change the Initial *CurrentTransformation*.

The operators of clause 23 which affect the *CurrentTransformation* are of two forms. The first form of operator computes a new *CurrentTransformation* relative to the existing *CurrentTransformation* and installs the result as the new *CurrentTransformation*. The second form of operator computes transformations as abstract entities which can either be appended to the *CurrentTransformation* using the **Concat** operator or installed as the Active *CurrentTransformation* using the **SetTrans** operator.

17.4 Notation

17.4.1 Literals

Literals are single tokens or sequences of tokens that stand for an object. Except for operator names, literals within this International Standard shall be represented using the syntax for token sequences in the Clear Text Content Interchange Format defined in 37.2. The representation of operator names is given below in 17.1.1.3.

EXAMPLE 1

```
0.5

2

[0 0 0]
```

17.4.2 Objects

An object is identified by both its type and its content. The content of a value is just the value itself. The content of a composite object is the set of values of which it is composed. The content may be denoted either by a literal for the value or by a name representing any of the possible values of the designated type. An object of type "*Type*" and whose content is denoted by "*Content*" is denoted by "< *Content: Type* >".

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642650 246

When the content is denoted by a name, which may be written in a mixture of upper and lower case letters, the name is italicized.

EXAMPLE 2

<n: Number >

<flag: Boolean >

<T: Transformation >

When the content is a specific value or composite object it is denoted by a literal for that value or composite object. Normally such literals are not italicized.

EXAMPLE 3

<0.5: *Number* >

<2: *Integer* >

<[0 0 0]: *Vector* >

Dictionary keys, although literal identifiers, are normally italicized to identify them as part of the SPDL language and to distinguish them form the English words used to describe SPDL. This includes the names of interpreter errors in *ErrorDict*.

When the type of a object is implied by the content and there are a number of such objects in close proximity, the content literal may stand alone, without its associated type, as in the last example above.

17.4.3 Object Types

An object type is denoted by a type name which is expressed by a string of italicized characters that begins with an uppercase letter and continues with lowercase letters, except that the initial letter of each word in a compound type name is an uppercase letter:

EXAMPLE 4

Number

Transformation

VectorReference

17.4.4 Abbreviated Forms

When it is not necessary to specify a specific object or give it a name, an object may be denoted by the name of the type of the object in non-italicized text:

EXAMPLE 5

Number

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642651 182

Transformation

VectorReference

When an object whose content is denoted by a name is referred to several times in the same subclause and the context provides enough information to avoid ambiguity, the object may be denoted solely by the name of the content

EXAMPLE 6

"*n*" and "*T*" may be used to make repeated references to "*<n: Number >*"

and "*<T: Transformation >*".

17.4.5 Sequences of objects

A sequence of objects is denoted by a parenthesized, comma-separated list of object representations.

EXAMPLE 7

(Integer, Integer, Integer)

Is a sequence of three objects all of type *Integer*.

A sequence of an indefinite number of objects is denoted by a parenthesized, comma-separated list of object representations in which the ellipsis, ". . .", is used.

EXAMPLE 8

($value_1$, $value_2$, . . . $value_n$)

Is a sequence of *n* objects named $value_1$ through $value_n$.

17.4.6 Operators

An operator is denoted by an operator name expressed as a string of boldface characters that begins with an uppercase letter and continues with lowercase letters, except that the initial letter of each word in a compound operator name is an uppercase letter. The names of operators are given in annex A.

EXAMPLE 9

Multiply

FindResource

ImageRasterElement

17.4.7 State and Graphics State Variables

A State Variable or a Graphics State Variable is denoted by its name which is expressed by a string of italicized characters that begins with an uppercase letter and continues with lowercase letters, except that the initial

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642652 019

letter of each word in a compound variable name is an uppercase letter. The names of all Graphics State Variables are compound names in which the initial word is "*Current*":

EXAMPLE 10

CurrentColor

CurrentTransformation

CurrentClippingRegion

17.4.8 Operand Stack

When it is necessary to represent information about the state of the top portion of the Operand Stack, that state is represented by showing the top *n* values on the Operand Stack as a vertical list. The top value on the Operand Stack is given as the first item in the list, the second value as the second item, and so forth. The format is shown below:

$<value_1: Any>$ operand on the top of the stack

$<value_2: Any>$ second value on the operand stack

. .

$<value_n: Any>$ n^{th} value on the operand stack

It is understood that an arbitrary number of additional values may be below the shown values on the Operand Stack Those values that are not shown are not affected or used by any operator that acts on the shown portion of the Operand Stack.

17.4.9 Operator descriptions

For each operator in this International Standard, the operator description specifies

— the number and types of operands which are popped from the Operand Stack at the beginning of the interpretation of the operator.

— the number and types of the results pushed onto the Operand Stack upon completion of the interpretation of the operator

— the relationships between the operands and the results.

— any other effects the interpretation of the operator has upon the Virtual Machine State.

— any effects upon the Current Page Image

The list of operands and the list of results are given as Operand Stack state representations.

The description of the effects of interpreting an operator may be given solely in terms of a textual description or the text may include a sequence of SPDL tokens whose interpretation would produce an equivalent effect. The notation

{ sequence-of-tokens }

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

stands for the result of interpreting the sequence of tokens within the curly braces. In this case, the tokens will be separated by spaces and will be represented by operator names or literals. Comments may be included within the sequence of tokens. Such comments are separated from the tokens by a percent sign (%) and continue to the end of the line.

18 Content Data Types

This Clause defines the types of objects that can occur in the processing of content. These types include some, but not all, of the types defined in Clause 6. Those clause 6 types which can occur are explicitly specified below. Only types defined or referenced in this Clause may occur in the processing of content values.

Each of the objects processed by the SPDL virtual machine has an associated object type. The structure of the type system is described in clause 6 . This type system was extended in clause 17 with the introduction of composite objects and object references. Values and composite objects are subclasses of the class of objects. Object references are a subclass of values.

Composite objects are containers for values, both simple values and object references. Composite objects are identified by object references. Thus, one composite object may refer to another. The composite object types are described in 18.2. Every composite object type implicitly defines a corresponding object reference type. No separate definition will be given for the object reference type that references a given composite object type.

There are some object reference types that refer to a subset of a composite object type. For these object references an explicit definition is given in 18.3 .

The subset of object types which have a literal representation in the Content Interchange Format is specified in clause 37 and clause 38 .

NOTE 83 Those types which do not have a literal representation in the Content Representation and Interchange Format are also listed in clause 37.

18.1 Value types

The value types include the simple value types and the object reference types. Some of the types of simple values used in Structure are also used in Content. Such value types are identified here by reference to the definition in clause 6 . The remaining simple values usable in Content are defined in this The object reference types are defined implicitly in 18.2 and explicitly in 18.3.

18.1.1 Any

The *Any* type is a super type (see clause 6) of all the content value types, both simple value types and object references. The set of values of the *Any* type consists of the union of the set of values all the types defined in 18.1, 18.3 and the implied object reference types for the composite objects defined in 18.2. Two values of the *Any* type are equal only if both values are of the same base type and they are equal within that base type.

18.1.2 Boolean

The *Boolean* type is defined in 6.1.

18.1.3 Cardinal

The *Cardinal* type is defined in 6.4.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642654 991

18.1.4 Dictionary Key

The *DictionaryKey* type is a super type of the *Identifier* type (see 18.1.5) and the *Cardinal* type. The set of values of the *DictionaryKey* type consists of the union of the set of values of the *Identifier* type and the set of values of the *Cardinal* type. Two values of the*DictionaryKey* type are equal if and only if they are both of the same base type and are equal as values of that type.

NOTE 84 As described in 18.2.4, Dictionaries contain key/value pairs. The *DictionaryKey* type defines the set of values which may be used as the name component of such a key/value pair.

18.1.5 Identifier

The *Identifier* type is a base type. The set of values of the *Identifier* type consists of a set of abstract values each of which corresponds to a sequence of octets where each octet represents an ISO 646 character encoding. Two values of the *Identifier* type are equal if and only if the octet strings corresponding to each value are equal.

A subset of the set of all possible Identifiers may be encoded directly in content (see the subtype *Name* in 18.1.8). An Identifier whose value does not conform to the restrictions placed on the *Name* type may be encoded as an *OctetStrings* and converted to an Identifier via execution of the **ConvertToIdentifier** operator.

18.1.6 Integer

The *Integer* type is defined in 6.3.

18.1.7 Mark

The *Mark* type is a base type. The set of values of the *Mark* type consists of a single value, the result of the **Mark** operator. This value may be used to delimit sequences of arguments on the operand stack.

18.1.8 Name

The *Name* type is a sub-type of the *Identifier* type. This type represents the subset of the *Identifier* type which may be encoded directly in content. The set of values of the *Name* type consists of those abstract values of *Identifier* type that correspond to an octet string with the following properties: the first octet of the value of the Name shall be the coding of an ISO 646-encoded character from the 26 letter Latin alphabet: "a", "b", "c", . . . , "z"; "A", "B", "C", . . "Z"; or a period (fullstop) ".". Each succeeding octet in the sequence may be chosen from the Latin alphabet letters, decimal digits "0", "1", "2", . . . , "9", the underbar "_", the hyphen "-", the colon ":", or the period (fullstop) ".".

In content, an object of *Name* type may be executable or literal. The processing of each is outlined in 17.1.3 and 17.1.4.

18.1.9 Null

The *Null* type is a base type. The set of values of the *Null* type consists of a single value, the result of the **Null** operator. This value is used to initialize the elements of Vectors at the time of their creation by the **MakeVector** operator.

18.1.10 Number

The *Number* type is defined in 6.7.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642655 828

18.1.11 Operator

The *Operator* type is a base type. The set of values of the *Operator* type consists of a set of values each of which, when executed, causes the virtual machine to perform a specific action or set of actions. Execution of a value of *Operator* type causes the interpretation of the Operator as outlined in 17.1.4 . The semantics of the values of *Operator* type are defined by this International Standard. Each value of *Operator* type is the value of a key/value pair in the standard *Dictionary*, *SystemDict*. The content of SPDL documents may not create values of *Operator* type, but may cause Identifiers to be bound to such values.

A value of *Operator* type is executed by the virtual machine when a Name bound to the object is encountered in content, or when the operand to the **Execute** operator is an object of *Operator* type or is an Identifier which is bound to an object of *Operator type*.

18.1.12 Real

The *Real* type is defined in 6.6.

18.1.13 SaveObject

The *SaveObject* type is a base type. The set of values of the *SaveObject* type consists of the results of the **SaveState** operator. Each such value has all the information necessary to allow the **RestoreState** operator to restore the Virtual Machine State at the time the **SaveState** operator was executed and the *SaveObject* was created.

18.1.14 StreamObject

The *StreamObject* type is a base type. The set of values of the *StreamObject* type consists of the results of the **Filter** operator or of the **FindResource** operator when used to find a Data Source resource. A value of the *StreamObject* type provides content-level access to data from Data Source resources or data that is output from filters (see clause 31).

18.2 Composite object types

The composite object types define containers which are identified by object references. A composite object shall not be used directly as an operand of a content operator; instead an object reference for the composite object shall be used. The composite object types are explicitly defined in this sub-clause. The definition of the corresponding object reference type is implicit; for each composite object type *X* there is an object reference type *XReference*.

NOTE 85 The composite objects are defined here only as part of the virtual machine model. Like the virtual machine model, they are presented in this International Standard only as a mechanism for defining the semantics of document content. It is not intended to dictate the actual implementation of a document content processor, nor to restrict the method by which document content semantics are realized in a particular implementation.

18.2.1 Dictionary

The *Dictionary* type is a composite object type. The *Dictionary* type consists of a set of containers each of which holds a set of key/value pairs. Each key/value pair shall consist of a value of the *DictionaryKey type*, and another value of the *Any type*. The first element of each key/value pair is called the *key*, and the second element is called the *value*. Within a single Dictionary, no two key/value pairs shall have the same key.

An object *<key: DictionaryKey>* is said to be *in* a Dictionary, or a Dictionary is said to *contain* the *<key: DictionaryKey>*, if there is an key/value pair in the Dictionary whose key is *<key: DictionaryKey>*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642656 764

If <*key: DictionaryKey*> is in a Dictionary, then the *value bound to key by Dictionary* is the value of the key/value pair whose key is *key*. If the Dictionary containing *key* is evident from context, then this may be called simply the *value bound to key* or the *value of key*.

Dictionaries are created by the **MakeDictionary** or the **MakeandStoreDictionary** operator during processing of document content or by the processing of a DICTIONARY RESOURCE SPECIFICATION in structure. Dictionaries are referenced explicitly via a DictionaryReference.

A Dictionary shall have a *capacity* which shall be the number of pairs the Dictionary can hold. When the Dictionary is created, an initial capacity shall be specified. If the Dictionary already has that many pairs and a pair with a key not already contained in the Dictionary is bound, then the capacity of the Dictionary shall be increased by at least one. The capacity of a Dictionary may be obtained using the **Capacity** operator. The number of pairs currently contained in the Dictionary may be obtained using the **EntriesUsed** operator.

An explicitly referenced Dictionary with an access attribute value of *ReadWrite* may be changed using the **Put** operator. If the top Dictionary on the Context Stack has an access attribute value of *ReadWrite*, it may be changed implicitly using the **Define** operator. In both cases, the value operand shall be bound to the key operand within the specified Dictionary.

Provided that there are no later changes with that same key, that value may be retrieved explicitly by using the **Get** operator with the key and a reference to the same Dictionary as operands or the **GetTest** operator may be used to test if the Dictionary contains that key.

If the Dictionary changed using the **Put** operator is on the Context Stack and no Dictionary higher on the Context Stack contains the same key, then the value may be retrieved implicitly using an executable Name, provided the key is a *Name*, or it may be retrieved explicitly using the **GetValue** operator with the key when the key could be any *DictionaryKey*. The **GetValueTest** operator may be used to test if the key is contained in a Dictionary on the Context Stack.

18.2.2 OctetString

The *OctetString* type is a composite object type. The *OctetString* type consists of a set of containers each of which holds an octet Each octet is a value of the *Cardinal type* in the range 0 to 255, inclusive.

NOTE 86 an octet is also known as an 8-bit byte.

Objects of the *OctetString* type are created using the **MakeString** operator. The **MakeString** operator takes a Cardinal as its operand which specifies the capacity of the *OctetString*, the number of octets it can hold. The capacity of an *OctetString* cannot be changed once the *OctetString* has been created. The **Capacity** operator returns the capacity of an *OctetString*

Given an object reference to an *OctetString*, the content of the *OctetString* can be changed an element at a time using the **Put** operator or a subsequence at a time using the **PutInterval** operator. The elements can be accessed individually using the **Get** operator or the **GetInterval** operator returns an *OctetStringReference* to a subsequence of an *OctetString*.

18.2.3 Path

The *Path* type is a composite object type. The *Path* type consists of a set of containers each of which holds a sequence of zero or more path segments. Each *path segment* begins with a *start point* created by the **SetPosition** operator (see 27.2.3) and consists of a sequence of zero or more path elements. A *path element* is one of:

– a line segment

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642657 6T0

— an arc of a circle or ellipse

— a Bezier curve

Path segments and path elements each have a start point and an end point, which are locations in RCS. Points are defined in 17.3 .

The start point of the first path element in each path segment is the start point of the path segment. The start point of each other path element in the path segment is the end point of the preceding path element.

Each path segment has an end point. If the path segment consists only of a start point, then the end point of that path segment is identical to its start point. If the path segment has one or more path elements, the end point of the path segment is the end point of the last path element in that path segment and the path segment is termed *directed*, where its *direction* is defined to be from its start point to its end point.

Each path has a start point, which is the start point of its first path segment. Each path has an end point, which is the end point of its last path segment. As a single exception, a path with no path segments is called a *null path*, and has no start point or end point.

NOTE 87 The primary use of Paths is in the construction of geometric graphic imaging elements and clipping regions.

18.2.4 Vector

The *Vector* type is a composite object type. The *Vector* type consists of a set of containers each of which holds a sequence of values, termed *elements*. Each value in the sequence is associated with an *index*. The index of each element is a Cardinal, between zero and *n-1* inclusive, where *n* is the number of elements in the sequence. The index specifies the position of the element in the sequence. The elements of a Vector are not required to be of the same type.

A Vector may be created using the **MakeVector** or the **MakeandStoreVector** operator. It is created with a fixed capacity, *n* above, that cannot be changed once the Vector is created. The **Capacity** operator returns the capacity of a Vector.

Given an object reference to a Vector, the content of the Vector can be changed an element at a time using the **Put** operator or a subsequence at a time using the **PutInterval** operator. The elements can be accessed individually using the **Get** operator, or the **GetInterval** operator returns a *VectorReference* to a subsequence of a Vector.

18.3 Object reference types

In addition to the object reference types defined implicitly with each composite object type, there are several object reference types that refer to a subset of the composite objects in a particular composite object type. These subsets are subtypes of the class of composite objects, but these subtypes will not be given type names of their own. Instead, they will be referred to by the object reference type that refers to the subtype.

18.3.1 Indexed Font

The *IndexedFont object reference type* represents the subset of the *DictionaryReference object type* which references Dictionaries containing the information necessary for the presentation of character text using a particular font.

NOTE 88 IndexedFonts may be constructed by use of the **DefineFont**, **PutWMode**, **ScaleFont**, and **TransformFont** operators in document content or, by use of the **FindFont** and **FindResource** operators, may be selected from the Set of Resources made available through the document structure. An IndexedFont may be used for the rendering of character text by assigning its value to the Graphics State Variable *CurrentFont*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642658 537

18.3.2 Glyph String

The *GlyphString* object reference type represents the subset of the *OctetStringReference* object type which is suitable for use as an operand to the **ShowString** operator.

NOTE 89 The primary use of *GlyphStrings* is for the efficient presentation of character text. The elements of the *GlyphString* are interpreted as a sequence of glyph indices, representing glyphs by reference to the Encoding Vector associated with the *CurrentFont*.

18.3.3 Procedure

The *Procedure* object reference type represents the subset of the *VectorReference object* type in which each VectorReference has an executability attribute that is executable.

NOTE 90 Procedures may be encoded in the content notation, or may be created by applying the **ConvertToExecutable** operator to *VectorReferences*. A Procedure is interpreted when a name bound to the Procedure is encountered in content as an executable Name, or a Procedure may be caused to be interpreted by the **Execute** operator. See 17.1.3 for more details.

18.3.4 Transformation

The *Transformation* object reference type is the subset of the *VectorReference* type for which the referenced Vector is of the form:

[a b c d e f]

Such a Vector represents a linear mapping from one Cartesian Coordinate System into another, as described below

All such mappings can be represented in the form:

$$x' = a \times x + c \times y + e \quad ,$$

$$y' = b \times x + d \times y + f$$

where a, b, c, d, e, and f are numbers, x and y are coordinates in the first Cartesian Coordinate System, and x' and y' are coordinates in the second Cartesian Coordinate System.

NOTE 91 Any such mapping where $a \times d - b \times c = 0$ will be degenerate and non-invertible.

A particular set of coefficients a, b, c, d, e, and f are said to be *associated with* a particular transformation, and are also said to *define* the particular transformation. The coefficients are dimensionless unless the associated transformation is the current value of *CurrentTransformation*, in which case the coefficients have units of millimeters.

These equations can also be expressed in the form of a matrix multiplication as

$$[\, x' \;\; y' \;\; 1 \,] = [\, x \;\; y \;\; 1 \,] \times \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix}$$

for this reason the matrix M given by

$$M = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix}$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

is said to be *associated with* and to *define* the transformation T_M which is given by the above equations.

18.4 Attributes

Some SPDL objects have one or two *attributes* associated with them. The two attributes are an *executability attribute*, which determines whether the object is treated as data to be pushed onto the operand stack or as an executable object, and an *access attribute*, which determines whether the value of the object may be explicitly read, written or executed by SPDL content operators.

18.4.1 Executability Attribute

The executability attribute is either *literal* or *executable*. Only objects of the *Identifier*, *Operator*, or *VectorReference* types have an executability attribute. The executability attribute of objects of the *Operator* type is always executable. The executability attribute of an object of the *Identifier* or *VectorReference*types may explicitly be made executable via execution of the **ConvertToExecutable** operator.

18.4.1.1 Identifiers

The executability attribute of identifiers created via execution of the **ConvertToIdentifier** operator is literal at the time of their creation. The executability attribute of Names encountered in the content representation is determined as specified in 17.1.3 and clause 37 and clause 38.

18.4.1.2 VectorReferences

The executability attribute of *VectorReferences* created by the **MakeVector** operator is literal; their executability attribute can be made executable via execution of the **ConvertToExecutable** operator. *VectorReferences* with an executability attribute of executable belong to the *Procedure* type.

The attribute applies to the VectorReference, rather than the Vector being referenced. This implies that two *VectorReferences*, each with a different executability attribute, may reference the same underlying Vector but be treated differently in content.

18.4.2 Access Attribute

The access attribute applies only to objects of the *Dictionary*, *OctetStringReference*, or *VectorReference* types. There are four access attributes:

— *ReadWrite* - An object referenced via an object reference with an access attribute of *ReadWrite* may be explicitly read or written in content by operators or it may be executed (if it is executable) by the interpreter. A composite object is written by any operator which modifies the contents of the composite object (see 17.1.1.1)

— *ReadOnly* - An object referenced via an object reference with an access attribute of *ReadOnly* may be explicitly read, but not explicitly written, in content by operators and it may be executed (if it is executable) by the interpreter.

— *ExecuteOnly* - An object referenced via an object reference with an access attribute of *ExecuteOnly* may not be explicitly read or written in content by operators, but may be executed (if it is executable).

— *NoAccess* - An object referenced via an object reference with an access attribute of *NoAccess* may not be explicitly read or written in content by operators, nor may it be executed.

New objects of the *Dictionary*, *OctetStringReference*, or *VectorReference types* created in content have an access attribute of *ReadWrite* unless otherwise specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

For objects of the *OctetStringReference* and *VectorReference* types, the attribute applies to the reference, rather than the object being referenced (see the discussion under Executability Attribute in 18.4.1.2). For objects of the *Dictionary* type, however, the attribute is directly associated with the Dictionary.

The access attribute of an object may be reduced by the operators **MakeReadOnly**, **MakeExecuteOnly** or **NoAccess**.

19 State Variables

Two of the components of the virtual machine are: the *Graphics State Variables* and the *State Variables*. The values of these variables form part of the Virtual Machine State. These variables are global in that they can always be referenced from content. The set of *Graphics State Variables* is the set of state variables whose values are used by imaging operators. The *State Variables* hold information about the state of the Content Processor. The following subclauses define each state variable, its type, and its initial value at the time the Content Processor is called.

NOTES

92 The values of these variables are used as implicit operands by many operators. The semantics of such use are specified in the definition of each particular operator. Each subclause below refers, in a NOTE, to the clauses which define these operators and their semantics.

93 Some variables can be set explicitly, some can be queried explicitly, and some can be both set and queried, by operators in content. See the referenced clauses for the existence of operators that perform these functions.

19.1 Current Black Generation

The *CurrentBlackGeneration* Graphics State Variable has a value of type *Procedure*. Its initial value is implementation-dependent

NOTE 94 The *CurrentBlackGeneration* specifies the function for generating the black component of DeviceCMYK color during a DeviceRGB to DeviceCMYK color conversion. Operators specific to the manipulation of *CurrentBlackGeneration* are defined in clause 34

19.2 Current Clipping Region

The *CurrentClippingRegion* Graphics State Variable describes a region of the page image. Its initial value is a rectangle containing the entire imageable region of the page image as specified in clause 16.

NOTE 95 The *CurrentClippingRegion* is a part of the imaging model. It determines the area of the medium which can be affected by imaging activities. It is defined in 17.2 3 and used throughout clause 24 to clause 29. Operators specific to the manipulation of *CurrentClippingRegion* are defined in clause 30.

19.3 Current Color

The *CurrentColor* Graphics State Variable has as its value the specification for the current color/pattern in the current color space. The type and number of component values for this Graphics State Variable depend on the *CurrentColorSpace* Graphics State Variable. See 34.3 through 34.5 for details. Its initial value is <0: *Number*> (denoting 100% black) in the default color space (see 19.5 below).

NOTE 96 The *CurrentColor* specifies the color to be rendered for all geometric graphics and character text imaging operators, throughout clause 24 to clause 27 and clause 29, and the color of bitmap raster graphics. The color may be a solid color or a bit mask pattern in the *CurrentColorSpace* or a pattern which selects its own color space. Changing the color space via the **SetColorSpace** operator establishes a new *CurrentColor*. This initial value of *CurrentColor* for each color space and operators specific to the manipulation of *CurrentColor*, are defined in clause 32 and clause 34.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642661 021

19.4 Current Color Rendering

The *CurrentColorRendering* Graphics State Variable has a value of type *DictionaryReference*. Its initial value is implementation-dependent.

NOTE 97 The *CurrentColorRendering* specifies the parameters (as a color rendering Dictionary) to the function which converts CIE-related color into device color. Operators specific to the manipulation of *CurrentColorRendering* are defined in clause 35.

19.5 Current Color Space

The *CurrentColorSpace* Graphics State Variable has a Color Space Object which is a value of type *VectorReference*. The initial value is a reference to a single-element Vector that is the result returned by **FindResource** when applied to an INTERNAL RESOURCE NAME that is declared to be bound to a Color Space resource (see 9.6.9) for which the Object Name is "*color-space-family*':*DeviceGray*".

NOTE 98 The *CurrentColorSpace* specifies the domain in which colors are specified, and the set of component values used in that specification. If the *CurrentColor* is a pattern, the referenced Vector specifies the color space underlying the pattern. Operators specific to the manipulation of *CurrentColorSpace* are defined in clause 32 and clause 34.

19.6 Current Dash Pattern

The *CurrentDashPattern* Graphics State Variable has two component values. a reference to a Vector of Numbers and a numerical offset. The initial values are a reference to an empty Vector and an offset of <0: *Number*>.

NOTE 99 The *CurrentDashPattern* specifies the dash feature generated along paths imaged by the **StrokePath** operator. Operators specific to the manipulation of this state variable are defined in clause 29 .

19.7 Current Font

The *CurrentFont* Graphics State Variable has a value of type *IndexedFont*. Its initial value is <*null: Null*>; if an operation which uses the *CurrentFont* is attempted before a value has been assigned to *CurrentFont*, **RaiseError** shall be invoked with *InvalidFont* as its operand. The operators **GetSelectedFont** and **GetRootFont** which only access the value will not invoke **RaiseError**.

NOTE 100 The *CurrentFont* specifies the *IndexedFont* to be used for all character text imaging operators; see clause 24 to clause 27. Operators specific to the manipulation of *CurrentFont* are also defined in clause 27 .

19.8 Current Halftone

The *CurrentHalftone* Graphics State Variable has a value of type *DictionaryReference*. Its initial value is implementation-dependent.

NOTE 101 The *CurrentHalftone* specifies the parameters (as a halftone Dictionary) to the halftoning function. Operators specific to the manipulation of *CurrentHalftone* are defined in Clause 35.

19.9 Current Miter Limit

The *CurrentMiterLimit* Graphics State Variable has a value of type *Number*. Its initial value is <10: *Number*>.

NOTE 102 The *CurrentMiterLimit* is used by the **StrokePath** operator to determine when a bevel join is used instead of a miter join; see clause 18. Operators specific to the manipulation of this state variable are also defined in clause 29.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642662 T68

19.10 Current Path

The *CurrentPath* Graphics State Variable is a Path. Its initial value is the null path All points in the *CurrentPath* are points in the Reference Coordinate System (see 17.3).

NOTE 103 The *CurrentPath* is used by operators that image Paths, and manipulated by operators that build Paths; see clause 29 .

19.11 Current Position

The *CurrentPosition* Graphics State Variable is a point in the Reference Coordinate System (see 17.3). The value of the *CurrentPosition* Graphics State Variable is undefined if and only if the value of the *CurrentPath* Graphics State Variable is the null path. Otherwise, the value of the *CurrentPosition* Graphics State Variable is end point of the *CurrentPath*.

NOTE 104 The *CurrentPosition* specifies the location, in RCS coordinates, where the beginning of a character text glyph sequence will be placed for imaging, or starting point of the next path element in a geometric graphics sequence. Operators specific to the manipulation of this state variable are defined in clause 27 and clause 29 .

19.12 Current *OverPrint*

The *CurrentOverPrint* Graphics State Variable has a value of type *Boolean*. Its initial value is false.

NOTE 105 the *CurrentOverPrint* Graphics State Variable is used to control painting when generating color separations, something that some presentation devices are able to do. When painting on one separation, false causes the corresponding areas in the other separations to be erased and true leaves them unchanged; see clause 34. This can be used to implement trapping, chokes , spreads and grip

19.13 Current Stroke Adjust

The *CurrentStrokeAdjust* Graphics State Variable has a value of type *Boolean*. Its initial value is implementation-dependent.

NOTE 106 When a stroke is drawn along a path via execution of the **StrokePath** operator, the scan conversion algorithms may produce lines of non-uniform thickness due to rasterization effects. The problem of non-uniform thickness lines is especially noticeable on low-resolution devices such as softcopy displays. If the value of *CurrentStrokeAdjust* is true, the stroke may be adjusted to produce lines of uniform thickness Operators specific to the manipulation of this state variable are defined in clause 29 .

19.14 Current Stroke End

The *CurrentStrokeEnd* Graphics State Variable has a value of type *Cardinal*. Its initial value is <0: *Cardinal*>.

NOTE 107 The *CurrentStrokeEnd* specifies the type of end feature applied to path ends imaged by the **StrokePath** operator. Operators specific to the manipulation of this state variable are defined in clause 29 .

19.15 Current Stroke Join

The *CurrentStrokeJoin* Graphics State Variable has a value of type *Cardinal*. Its initial value is <0: *Cardinal*>.

NOTE 108 The *CurrentStrokeJoin* specifies the type of join between non-tangential path elements imaged by the **StrokePath** operator. Operators specific to the manipulation of this state variable are defined in clause 29.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642663 9T4

19.16 Current Stroke Width

The *CurrentStrokeWidth* Graphics State Variable has a value of type *Real*. Its initial value is <1.0: *Real*>.

NOTE 109 The *CurrentStrokeWidth* specifies, in UCS units, the width of lines imaged by the **StrokePath** operator. Operators specific to the manipulation of this state variable are defined in clause 29.

19.17 Current Transformation

The *CurrentTransformation* Graphics State Variable has a value of type *Transformation*. Its initial value is the Transformation which maps the axes of the UCS to the corresponding axes of the page image RCS as described in clause 16 , and maps 1 unit in the UCS to 1 unit (1 millimeter) in the RCS.

NOTE 110 The *CurrentTransformation* specifies the mapping to be applied to coordinates in the UCS to transform them into coordinates in the RCS. This mapping is defined in 17.3, and used throughout clause 24 to clause 33. Operators specific to the manipulation of *CurrentTransformation* are defined in clause 23.

19.18 Current UnderColor Removal

The *CurrentUnderColorRemoval* Graphics State Variable has a value of type *Procedure*. Its initial value is implementation-dependent.

NOTE 111 The *CurrentUnderColorRemoval* specifies the function for compensating for black generation by adjusting CMY values during a DeviceRGB to DeviceCMYK color conversion. Operators specific to the manipulation of *CurrentUnderColorRemoval* are defined in clause 35.

19.19 Device Description Dictionary

The *DeviceDescription* State Variable has a value of type *DictionaryReference*. The contents of this *ReadOnly* Dictionary include at least the key/value pairs described in the following subclauses. The initial values of these mandatory key/value pairs, as well as the keys and values of any additional key/value pairs, are implementation-dependent.

NOTE 112 The contents of the *DeviceDescription* make available to document content certain aspects of the current state of the presentation device. The current value of the *DeviceDescription* DictionaryReference is obtained via execution of the **GetDeviceDescription** operator described in clause 22.

19.19.1 Process Color Class

The *ProcessColorClass* key has a value of type *Name*. This Name value specifies the colorant model used for rendering process colors in the device. Each *ProcessColorClass* determines the names of the colorants that are used as keys in a type 5 Halftone Dictionary (see clause 35). These are the process colorant names used to select component Halftone Dictionaries in a type 5 Halftone Dictionary and to control the production of separations. The valid values of *ProcessColorClass* are as shown in table 28.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 28

Value	ProcessColorClass
DeviceGray	The presentation device has a single process colorant whose name is Black
DeviceKX	The presentation device has two process colorants whose names are Black and Default
DeviceRGB	The presentation device has three process colorants whose names are Red, Green and Blue
DeviceRGBK	The presentation device has four process colorants whose names are Red, Green, Blue and Black
DeviceCMY	The presentation device has three process colorants whose names are Cyan, Magenta and Yellow
DeviceCMYK	The presentation device has four process colorants whose names are Cyan, Magenta, Yellow and Black

19.19.2 Current Resolution

The current resolution is reported in UCS units per pel. The UCS may be rotated with respect to RCS and the corresponding X and Y axes of the medium. Since the resolution of the underlying device may differ in the X and Y directions of the medium, the current resolution is given in terms of the image in UCS of two resolution vectors defined in RCS The term "vector" used here refers to its usage in mathematics and not the type Vector defined in this International Standard. These two, the X and Y resolution vectors are oriented parallel to the X and Y axes of RCS, respectively. The magnitudes of the X and Y resolution vectors are the number of RCS units per device pel in the X and Y directions of RCS, respectively Within the *DeviceDescription* Dictionary, the key *CurrentResolution* is bound to a reference to a Vector of four Numbers. The first two are the X and Y coordinates of the X resolution vector in UCS and the last two are the X and Y coordinates of the Y resolution vector.

NOTE 113 If the value of the first Number of *CurrentResolution* is $cXrX$, the value of the second Number of *CurrentResolution* is $cXrY$, and the value of the third Number of *CurrentResolution* is $cYrX$, then a unit step along the X axis of UCS moves $1/cXrX$ pels in the X direction of RCS and $1/cYrX$ pels in the Y direction of RCS. Moving $(cXrX,cXrY)$ units in UCS moves one pel in the direction of the X axis in RCS. This information allows a generator of SPDL to position image elements an integral number of device pels apart so they will render consistently with one another.

19.19.3 Current Medium Size

The *CurrentMediumSizeX* and *CurrentMediumSizeY* keys each have a value of type Number, and make available to document content the actual size in millimeters of the medium, along the x- and y-axes of the RCS.

19.19.4 Current Imageable Region

The keys *CurrentImageMinX*, *CurrentImageMaxX*, *CurrentImageMinY*, and *CurrentImageMaxY* each have a value of type *VectorReference* which corresponds to a corner point of the maximum rectangular region of the page image on which imaging is possible. Each Vector has two elements of type Number which are the X and Y coordinates, in UCS, of the corresponding corner point of the imageable region, in RCS.

NOTE 114 The boundary of the imageable region is typically determined in an implementation of SPDL by taking into account characteristics of the presentation device and the medium upon which the page image is rendered. The coordinates of the imageable region may depend on the positioning of the page image on that medium In particular, presentation parameters, such as *x-image-shift*, *y-image-shift* and *plex*, may affect the coordinates of the imageable region.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642665 777

20 Arithmetic and Logic Operators

This next three clauses specify operators which are not directly involved in manipulating or imaging page elements and related objects. This includes:

— arithmetic and logic operators;

— operators for manipulating values on the Operand Stack;

— operators for manipulating Dictionaries, OctetStrings, and Vectors;

— operators for flow control and manipulation of Procedures;

— operators for saving and restoring Virtual Machine State;

— operators for object type conversion and resource manipulation.

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and exception handling are defined in clause 36. In addition to these operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6.2.

NOTE 115 It is important to emphasize that SPDL specifies only the effect to be achieved, and not the mechanism which must be used to produce that result. While the definitions of this clause may, as an explanatory convenience, call for specific Operand Stack manipulations and object manipulations in a specific order, it is likely that the actual implementations will achieve the desired effect of a given operator in a different manner

The arithmetic operators described below perform their operations in accordance with the rules for binary floating point arithmetic set forth in IEEE 754 using the default values where options, such as traps, are specified in IEEE 754.

In the specifications in this clause, an operator which has two or more operands of type *Number* shall accept both *Integer* and *Real* values within the range of the operator. If one or more operands are *Integer*s and some operand is a *Real* then the *Integer* operands will be converted to *Real* to perform the operation.

20.1 AbsoluteValue

The **AbsoluteValue** operator takes a single operand,

<x: Number>

and returns a single result

<y: Number>

where *y* shall be the absolute value of *x*. If the operand *x* is of type *Integer*, the result *y* shall be of type *Integer*; otherwise *y* shall be of type *Real*.

20.2 Add

The **Add** operator takes two operands,

<y: Number>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642666 603

<x: Number>

and returns a single result

<z: Number>

where $z = x + y$. If both operands are of type *Integer*, the result z shall be of type *Integer* if z is representable as a value of type *Integer*; otherwise z shall be of type *Real*.

20.3 ArcTangent

The **ArcTangent** operator takes two operands,

<x: Number>

<y: Number>

and returns a single result

<a: Real>

where a shall be the angle between 0 and 360 in degrees whose tangent is y / x. The sign of the operands determine the quadrant in which the result will lie as shown in table 29.

Table 29

x	y	Result
Positive	Positive	Positive X, Positive Y quadrant
Positive	Negative	Positive X, Negative Y quadrant
Negative	Positive	Negative X, Positive Y quadrant
Negative	Negative	Negative X, Negative Y quadrant
Positive	Zero	0.0
Negative	Zero	180.0
Zero	Positive	90.0
Zero	Negative	270 0
Zero	Zero	**RaiseError** shall be invoked with *UndefinedResult* as its operand.

20.4 And

The **And** operator takes two operands,

<y: Boolean or Integer>

<x: Boolean or Integer>

and returns a single result

<z: Boolean or Integer>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642667 54T

Both operands must be of the same type. If both operands are of type *Boolean*, *z* shall be also of type *Boolean*, and the value of *z* shall be the Boolean intersection of the values of *x* and *y*. If both operands are of type *Integer*, *z* shall be also of type *Integer*, and the value of *z* shall be the bitwise *and* of the binary representations of *x* and *y* in two's complement representation.

20.5 Ceiling

The **Ceiling** operator takes a single operand,

<x: Number>

and returns a single result

<y: Number>

where *y* shall be the least integer value greater than or equal to *x* If the operand is of type *Integer*, *y* shall be of type *Integer*. If the operand is of type *Real*, *y* shall be of type *Real*.

20.6 Cosine

The **Cosine** operator takes one operand,

<a: Number>

and returns a single result

<c: Real>

where *a* is an angle expressed in degrees, and *c* shall be the cosine of that angle.

20.7 Divide

The **Divide** operator takes two operands,

<y: Number>

<x: Number>

and returns a single result

<z: Real>

where *z* shall be the result of dividing *x* by *y*. If *y* is zero, **RaiseError** shall be invoked with *UndefinedResult* as its operand.

20.8 Equal

The **Equal** operator takes two operands,

<y: Any>

<x: Any>

and returns a single result

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

<*z: Boolean*>

where *z* shall be true if *x* and *y* are both of the same type and their values are equal; and *z* shall be false otherwise — exceptions to this general rule are listed below.

If *x* and *y* are both of type *Number*, they may be compared even if one operand is of type *Integer* and the other is of type *Real*. In this case, the operand of type *Integer* shall be converted to type *Real* and the two operands shall be compared as values of type *Real*.

If *x* and *y* are both object references, then *x* and *y* are considered to be equal if they both reference the same object.

Values of type *Identifier* and *OctetStringReference* may be compared; in this case the *OctetString* that corresponds to the value of type *Identifier* and the referenced *OctetString* are compared as simple sequences of octets.

The attributes (18.4) associated with the values are not considered in determining the equality of values.

20.9 Exponentiate

The **Exponentiate** operator takes two operands,

<*y: Number*>

<*x: Number*>

and returns a single result

<*z: Real*>

where the value of *z* shall be *x* raised to the *y* power. If x is negative and y has a fractional component, **RaiseError** shall be invoked with *UndefinedResult* as its operand.

20.10 False

The **False** operator takes no operands and returns a single result

<false: *Boolean*>

which shall be a Boolean value which shall be false

20.11 Floor

The **Floor** operator takes a single operand,

<*x: Number*>

and returns a single result

<*y: Number*>

where *y* shall be the greatest integer value less than or equal to *x*. If the operand is of type *Integer*, *y* shall be of type *Integer*. If the operand is of type *Real*, *y* shall be of type *Real*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642669 312

20.12 GreaterOrEqual

The **GreaterOrEqual** operator takes two operands,

<*y: Number*>

<*x: Number*>

and returns a single result

<*z: Boolean*>

where *z* shall be the result of {*x y* **LessThan Not**}.

20.13 GreaterThan

The **GreaterThan** operator takes two operands,

<*y: Number*>

<*x: Number*>

and returns a single result

<*z: Boolean*>

where *z* shall be true if $x > y$, and *z* shall be false otherwise.

20.14 IntegerDivide

The **IntegerDivide** operator takes two operands,

<*y: Integer*>

<*x: Integer*>

and returns a single result

<*z: Integer*>

where *z* shall be the result of {*x y* **Divide Truncate**}. If *y* is zero, **RaiseError** shall be invoked with *UndefinedResult* as its operand.

20.15 LessOrEqual

The **LessOrEqual** operator takes two operands,

<*y: Number*>

<*x: Number*>

and returns a single result

<*z: Boolean*>

where *z* shall be the result of {*x y* **GreaterThan Not** Not}.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642670 034

20.16 LessThan

The **LessThan** operator takes two operands,

<y: Number>

<x: Number>

and returns a single result

<z: Boolean>

where *z* shall be true if $x < y$, and *z* shall be false otherwise.

20.17 Logarithm

The **Logarithm** operator takes one operand,

<n: Number>

and returns a single result

<log: Real>

where *log* shall be the common logarithm (base 10) of *n*.

20.18 LogicalShift

The **LogicalShift** operator takes two operands,

<s: Integer>

<m: Integer>

and returns a single result

<n. Integer>

where *m* is the value whose binary representation is to be logically shifted, *s* specifies the number of bits to shift and the direction of the shift, and *n* shall be the resultant shifted value. If *s* is positive, a left-shift of *s* bits shall be performed on m; if *s* is negative a right-shift shall be performed on *m*. Bits shifted in shall have a zero value.

20.19 Multiply

The **Multiply** operator takes two operands,

<y: Number>

<x: Number>

and returns a single result

<z: Number>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642671 T70

where *z* is the result of multiplying *x* by *y*. If both operands are of type *Integer*, the result *z* shall be of type *Integer* if *z* is representable as a value of type *Integer* and a value of type *Real*, otherwise; otherwise *z* shall be of type *Real*.

20.20 NaturalLogarithm

The **NaturalLogarithm** operator takes one operand,

<n: Number>

and returns a single result

<ln: Real>

where *ln* shall be the natural logarithm (the logarithm base *e*) of *n*.

20.21 Negate

The **Negate** operator takes a single operand,

<x: Number>

and returns a single result

<y: Number>

where *y* = -*x*. If the operand *x* is of type *Integer*, the result *y* shall be of type *Integer*; otherwise *y* shall be of type *Real*.

20.22 Not

The **Not** operator takes a single operand,

<x: Boolean or Integer>

and returns a single result

<z: Boolean or Integer>

If the operand is of type *Boolean*, *z* shall also be of type *Boolean*; the value of *z* shall be true if the value of *x* is false, and *z* shall be false otherwise. If the operand is of type *Integer*, *z* shall also be of type *Integer*, and the value of *z* shall be the one's complement of the binary representation of *x*.

20.23 NotEqual

The **NotEqual** operator takes two operands,

<y: Any>

<x: Any>

and returns a single result

<z: Boolean>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642672 907

where z is the result of {x y **Equal Not**}.

20.24 Null

The **Null** operator takes no operands and returns a single result

<null: Null>

which shall be the value of type *Null*.

20.25 Or

The **Or** operator takes two operands,

<y: Boolean or Integer>

<x: Boolean or Integer>

and returns a single result

<z: Boolean or Integer>

Both operands must be of the same type. If both operands are of type *Boolean*, z shall also be of type *Boolean*, and the value of z shall be the Boolean union of the values of x and y. If both operands are of type *Integer*, z shall also be of type *Integer*, and the value of z shall be the bitwise 'inclusive or' of the binary representations of x and y

20.26 Rand

The **Rand** operator takes no operands and returns a single result

<rand. Real>

where the returned value shall be a pseudo random number between 0 and 1 inclusive.

20.27 RandSetState

The **RandSetState** operator takes a single operand

<seed: Integer>

and returns no results. The effect shall be to initialize the state of the random number generator. A given seed will always result in the same set of numbers returned from the Rand operator in the same order on a given implementation. A given seed may not produce the same set of numbers across implementations. The seed shall be automatically set to a random value at the beginning of each high-level structure (DOCUMENT, PAGESET, PAGE, or PICTURE)

20.28 Remainder

The **Remainder** operator takes two operands,

<y: Integer>

<x: Integer>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642673 843

and returns a single result

<z: Integer>

where z shall be the remainder that results from the integer division of x by y. The sign of z shall be the same as the sign of x. If y is zero, **RaiseError** shall be invoked with *UndefinedResult* as its operand.

{x y **Remainder**}

shall be equivalent to

{x y x y **IntegerDivide Multiply Subtract**}

20.29 Round

The **Round** operator takes a single operand,

<x: Number>

and returns a single result

<y: Integer>

where y shall be the integer value nearest to x. If x is equally close to two integer values then y shall be the larger of the two.

20.30 Sine

The **Sine** operator takes one operand,

<a: Number>

and returns a single result

<s: Real>

where a is an angle expressed in degrees, and s shall be the sine of that angle.

20.31 SquareRoot

The **SquareRoot** operator takes one operand,

<x: Number>

and returns a single result

<y: Real>

where x shall be non-negative, and the result y shall be the non-negative square root of x.

20.32 Subtract

The **Subtract** operator takes two operands,

<y: Number>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

<x: Number>

and returns a single result

<z: Number>

where $z = x - y$. If both operands are of type *Integer*, the result z shall be of type *Integer* if z is representable as a value of type *Integer*; otherwise z shall be of type *Real*.

20.33 True

The **True** operator takes no operands and returns a single result

<true: *Boolean*>

which shall be a Boolean value which shall be true.

20.34 Truncate

The **Truncate** operator takes a single operand,

<x: *Number*>

and returns a single result

<y: *Number*>

If x is greater than or equal to zero, then y shall be the greatest integer value less than or equal to x. If x is less than zero, then y shall be the least integer value greater than or equal to x. If the operand is of type *Integer*, y shall be of type *Integer*. If the operand is of type *Real*, y shall be of type *Real*.

20.35 Xor

The **Xor** operator takes two operands,

<y: *Boolean or Integer*>

<x: *Boolean or Integer*>

and returns a single result

<z: *Boolean or Integer*>

Both operands must be of the same type. If both operands are of type *Boolean*, z shall also be of type *Boolean*; the value of z shall be true if the values of x and y are different, and z shall be false otherwise. If both operands are of type *Integer*, z shall also be of type *Integer*, and the value of z shall be the bitwise 'exclusive or' of the binary representations of x and y.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642675 616

21 Stack and Composite Object Operators

21.1 Operators for Manipulating Values on the Operand Stack

21.1.1 ClearStack

The **ClearStack** operator takes no operands and returns no results. The effect of the **ClearStack** operator is to remove all values from the stack.

21.1.2 ClearToMark

The **ClearToMark** operator takes no operands and returns no results. The effect of the **ClearToMark** operator is to remove from the stack all values above the highest Mark, and to remove the Mark as well. If there is no Mark object on the stack, **RaiseError** shall be invoked with *UnmatchedMark* as its operand.

21.1.3 ConvertToExecutable

The **ConvertToExecutable** operator takes a single operand,

<x: VectorReference or Identifier>

and returns a single result

<x: Procedure or Identifier>

where the result *x* has its executability attribute set to executable (see 18.4.1 for a discussion of the executability attribute) VectorReferences with an executability attribute of executable are known by the type *Procedure*.

21.1.4 ConvertToIdentifier

The **ConvertToIdentifier** operator takes a single operand,

<x: Name or OctetStringReference>

and returns a single result

<id: Identifier>

If the operand is of type *Name*, it is simply returned.

If the operand is of type *OctetStringReference*, the octets of the referenced *OctetString* are interpreted as a sequence of ISO 646-encoded characters and the result is the unique Identifier that corresponds to this sequence of octets (see 18.1.5).

21.1.5 ConvertToInteger

The **ConvertToInteger** operator takes a single operand,

<x: Number or OctetStringReference>

and returns a single result

<int: Integer>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642676 552

If the operand is of type *Integer*, it is simply returned.

If the operand is of type *Real*, its value is truncated as though by the **Truncate** operator and the resultant integer value returned

If the operand is of type *OctetStringReference*, each octet of the referenced OctetString is interpreted as an ISO 646-encoded character. The characters are interpreted as a number as if encountered in clear text content; the resultant Real or Integer operand is then interpreted as above. If the OctetString does not correctly encode a Real or Integer as specified in the Content Representation and Interchange Format, **RaiseError** shall be invoked with *SyntaxError* as its operand.

21.1.6 ConvertToReal

The **ConvertToReal** operator takes a single operand,

<x: Number or OctetStringReference>

and returns a single result

<real. Real>

If the operand is of type *Integer*, it is converted to a Real and returned.

If the operand is of type *Real*, it is simply returned.

If the operand is of type *OctetStringReference*, each octet of the referenced OctetString is interpreted as an ISO 646-encoded character The characters are interpreted as a number as if encountered in clear text content; the resultant Real or Integer operand is then interpreted as above. If the OctetString does not correctly encode a Real or Integer as specified in the Content Representation and Interchange Format, **RaiseError** shall be invoked with *SyntaxError* as its operand.

21.1.7 ConvertToString

The **ConvertToString** operator takes two operands,

<S: OctetStringReference> <x: Any>

and returns a single result

<newS. OctetStringReference>

The **ConvertToString** operator overwrites the operand *S* with an OctetString representation of the operand *x*; the return value *newS* is a new OctetStringReference that references the portion of the operand *S* that was overwritten

If the operand *x* is of type *Number*, the contents of *newS* are such that the characters, if encountered in the Clear Text Representation and Interchange Format for content, would result in the value represented by *x* being pushed onto the operand stack Operands of type *Real* are converted to the exponential representation described in 37.2.3.9 using a lower case "e"

If the operand is of type *Identifier*, the contents of *newS* are an OctetString containing the sequence of octets that corresponds to the Identifier.

If the operand is of type *Operator*, the Identifier to which the Operator is bound in *SystemDict* is converted to an OctetString as described above.

If the operand is of type *OctetStringReference*, the referenced OctetString is copied into S.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642677 499

If the operand is of type *Boolean*, the contents of *newS* are the OctetString true or the OctetString false, as appropriate to the value of *x*.

If the operand is of any other type, the contents of *newS* is the OctetString "--nostringval--".

21.1.8 Copy

21.1.8.1 Copy for *n* operand stack elements

The **Copy** operator takes $n+1$ operands, where n is the value on the top of the operand stack:

<n: *Cardinal*>

<x_{n-1}: *Any*>

. . .

<x_1: *Any*>

<x_0: *Any*>

and returns $2n$ results

<x_{n-1}: *Any*>

. . .

<x_1: *Any*>

<x_0: *Any*>

<x_{n-1}: *Any*>

. . .

<x_1: *Any*>

<x_0: *Any*>

The top operand n is popped to determine the number of values to copy, then the remaining n operands are popped and pushed back onto the stack, and finally the same n elements are pushed onto the stack again in the same order.

21.1.8.2 Copy for Vectors, Dictionaries, and OctetStrings

The **Copy** operator takes two operands

<x_1: *VectorReference, DictionaryReference, or OctetStringReference*>

<x_0: *VectorReference, DictionaryReference, or OctetStringReference*>

and returns a single result

<x_2: *VectorReference, DictionaryReference, or OctetStringReference*>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642678 325

Both operands shall be of the same type. All elements of the object referenced by x_0 are copied into the object referenced by x_1. The object referenced by x_1 shall be at least large enough to contain all the elements of the object referenced by x_0.

If the operands are DictionaryReferences, the Dictionary referenced by $x_1$shall be empty, and the returned value is the DictionaryReference operand x_1.

If the operands are OctetStringReferences or VectorReferences, any elements of x_1 not overwritten by elements from x_0 remain unaffected, and the returned value is a OctetStringReference or VectorReference to a newly-created OctetString or Vector whose length is the same as the length of the object referenced by x_0, and whose contents are identical to those of x_0

21.1.9 Count

The **Count** operator takes no operands and returns a single result

<n: Cardinal>

where *n* is the number of values on the stack.

21.1.10 CountToMark

The **CountToMark** operator takes no operands and returns a single result

<n: Cardinal>

where *n* is the number of values on the stack above the highest Mark. If there is no Mark object on the stack, **RaiseError** shall be invoked with *UnmatchedMark* as its operand.

21.1.11 Dup

The **Dup** operator takes a single operand,

<x: Any>

and returns two results

<x: Any>

<x: Any>

The top element of the stack is duplicated with no other effects.

21.1.12 Exchange

The **Exchange** operator takes two operands,

<y: Any>

<x. Any>

and returns two results

<x: Any>

<y: Any>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642679 261

The top two elements of the stack are exchanged with no other effects

21.1.13 Index

The **Index** operator takes $n+2$ operands, where n is the value on the top of the Operand Stack:

<n: *Cardinal*>

<x_n: *Any*>

<x_{n-1}: *Any*>

<x_{n-2}: *Any*>

. . .

<x_1: *Any*>

<x_0: *Any*>

and returns $n+1$ results:

<x_0: *Any*>

<x_n: *Any*>

<x_{n-1}: *Any*>

<x_{n-2}: *Any*>

. . .

<x_1: *Any*>

<x_0: *Any*>

The result is produced by popping n off the stack, then find the nth element down (counting the top element of the stack as the 0th element) on the stack and copy it onto the top of the stack.

NOTE 116 the expression {<x_0: *Any*> <0: *Cardinal*> **Index** } is equivalent to {<x_0: *Any*> **Dup** }.

21.1.14 Mark

The **Mark** operator takes no operands and returns a single result

<*mark: Mark*>

The only effect of the **Mark** operator is to push a *Mark* object on the stack.

21.1.15 Pop

The **Pop** operator takes a single operand,

<*x: Any*>

and returns no results. The only effect is the removal of the top element from the stack.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642680 T83

21.1.16 Roll

The **Roll** operator takes $n+2$ operands, where n is the value on the top of the Operand Stack:

<m: *Integer*>

<n: *Cardinal*>

<x_{n-1}: *Any*>

<x_{n-2}: *Any*>

. .

<x_1: *Any*>

<x_0: *Any*>

and returns n results. The top two operands m and n are popped to determine the number of values to shift (n), direction of shift (sign of m), and the number of positions by which to shift them (m). Then the remaining n operands are popped, a circular shift of these n values is performed, and the shifted sequence of values is pushed back onto the stack.

If the value of m is positive, shifting by one position is equivalent to removing the top element of the sequence of values, inserting it at the bottom of the sequence, and moving all intervening values one level higher in the stack:

<x_{n-2}: *Any*>

<x_1: *Any*>

<x_0: *Any*>

<x_{n-1}: *Any*>

If the value of m is negative, shifting by one position is equivalent to removing the bottom element of the sequence of values, moving all values above it one level lower in the stack, and pushing the removed element onto the top of the stack:

<x_0: *Any*>

<x_{n-1}: *Any*>

<x_{n-2}: *Any*>

.

<x_1: *Any*>

Shifting by m is equivalent to performing the shift by one position m times.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642681 91T

21.1.17 Type

The **Type** operator takes a single operand,

<*x: Any*>

and returns a single result

<*t: Identifier*>

where *t* identifies the base type of the operand *x* and is one of the following:

— *Boolean*

— *Dictionary*

— *Identifier*

— *Integer*

— *Mark*

— *Null*

— *OctetString*

— *Operator*

— *Path*

— *Real*

— *SaveObject*

— *StreamObject*

— *Vector*

21.2 Operators for Manipulating Dictionaries, Vectors, and OctetStrings

21.2.1 AnchorSearch

The **AnchorSearch** operator takes two operands

<*search: OctetStringReference*>

<*string: OctetStringReference*>

and returns either two or three results, depending on whether or not the initial substring of the OctetString referenced by *string* matches the OctetString referenced by *search*. If the initial substring of the OctetString referenced by *string* does match the search string, **AnchorSearch** returns three results

<true: *Boolean*>

<*match: OctetStringReference*>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642682 856

<*post: OctetStringReference*>

where *match* is a reference to the initial substring of the OctetString referenced by *string* that was matched, and *post* is a reference to the unmatched remainder of the OctetString referenced by *string*. If the search string does not match the initial substring of the *OctetString* referenced by *string*, **AnchorSearch** returns two results

<false: *Boolean*>

<*string: OctetStringReference*>

where *string* is a reference to the original OctetString being searched.

NOTE 117 In either case, the top element of the stack after the execution of **AnchorSearch** is a Boolean. The value of the Boolean is true if the search string matches the initial substring of the string being searched, and false otherwise.

21.2.2 Capacity

The **Capacity** operator takes a single operand

<*x: VectorReference, DictionaryReference, or OctetStringReference*>

and returns a single result

<*capacity: Cardinal*>

If *x* is of type *VectorReference* or *OctetStringReference*, the returned value is the number of elements, the capacity, of the object referenced by *x*.

If *x* is of type *DictionaryReference*, the returned object is the current total number of key/value pairs, the capacity, that can be stored in the Dictionary referenced by x.

21.2.3 ContextStack

The **ContextStack** operator takes a single operand

<*V: VectorReference*>

and returns a single result

<*V: VectorReference*>

The **ContextStack** operator enumerates the Dictionaries on the context stack into the Vector referenced by the operand VectorReference *V*. If *n* is the number of Dictionaries on the context stack, the returned VectorReference *V* is a reference to the subvector of the original Vector containing *n* elements, each of which is of type *DictionaryReference*. The element of the Vector referenced by the returned VectorReference *V* whose index is 0 contains a reference to the bottom most Dictionary on the context stack, and the element whose index is *n-1* contains a reference to the topmost Dictionary on the context stack. If the length of the Vector referenced by the operand VectorReference *V* is less than the depth of the context stack, **RaiseError** shall be invoked with *RangeCheck* as its operand.

21.2.4 Define

The **Define** operator takes two operands

<*new: Any*>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642683 792

<key: DictionaryKey>

and returns no results. The **Define** operator modifies the topmost Dictionary on the context stack. The Dictionary is searched for any key/value pair for which the key is equal to *key*. If any such pair is found, the value to which the name is bound is replaced by *new*

If no key/value pair is found whose key is equal to *key*, the key/value pair (*key*, *new*) is added to the Dictionary. If the Dictionary currently has as many pairs as its capacity, the capacity is increased by at least one. If this would increase the capacity beyond the implementation limit on dictionary size, then **RaiseError** is invoked with *LimitCheck* as its operand.

NOTE 118 Names bound in Dictionaries may be used as *executable Names* in content, as described in 17.1.3.

21.2.5 EntriesUsed

The **EntriesUsed** operator takes a single operand

<x: DictionaryReference>

and returns a single result

<entries: Cardinal>

where the returned value is the number of key/value pairs currently defined in the Dictionary referenced by *x* (as opposed to the current capacity of the Dictionary).

21.2.6 Get

The **Get** operator takes two operands

<key: Cardinal or DictionaryKey>

<x: VectorReference, DictionaryReference, or OctetStringReference>

and returns a single result

<value: Any>

If *x* is a *VectorReference* or OctetStringReference, *key* shall be of type *Cardinal*, and the value returned is the element of the object referenced by *x* whose index is *key*. If *key* is greater than or equal to the length of the object, or if it is negative, **RaiseError** shall be invoked with *RangeCheck* as its operand.

If *x* is a DictionaryReference the value returned is the value bound to *key* in the Dictionary referenced by *x*. If *key* is not in the Dictionary, **RaiseError** shall be invoked with *UndefinedKey* as its operand.

21.2.7 GetCurrentDictionary

The **GetCurrentDictionary** operator takes no operands and returns a single result

<DictRef: DictionaryReference>

The returned *DictionaryReference* is a reference to the Dictionary on the top of the current Context Stack. The Context Stack is unchanged by this operation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642684 629

21.2.8 GetInterval

The **GetInterval** operator takes three operands

<count: Cardinal>

<index: Cardinal>

<x: VectorReference or OctetStringReference>

and returns a single result

<interval: VectorReference or OctetStringReference>

The *interval* returned is of the same type as the operand *x*, the object it references is of length *count*, and its elements are composed of the subsequence of the object referenced by *x* beginning with the element whose index is *index* and continuing for *count* elements

If *index* is not a valid index into the original Vector or OctetString, or if *index*+*count* is greater than the length of the original Vector or OctetString, **RaiseError** shall be invoked with *RangeCheck* as its operand.

21.2.9 GetTest

The **GetTest** operator takes two operands

<key: DictionaryKey>

<DictRef: DictionaryReference>

and returns a single result

<found: Boolean>

GetTest returns a Boolean value of true if there exists a key/value pair in the Dictionary referenced by *DictRef* for which the key is *key*, and returns a value of false otherwise

21.2.10 GetValue

The **GetValue** operator takes a single operand

<key: DictionaryKey>

and returns a single result

<value: Any>

The value returned is determined by the context stack, in the context of interpretation. The Dictionaries on the context stack are examined in order, starting with the top Dictionary on the context stack. The topmost Dictionary which contains the DictionaryKey *key* will be used to determine the value bound to *key* in that Dictionary. This value will be returned on the operand stack. If *key* is not contained in any of the Dictionaries on the context stack, **RaiseError** shall be invoked with *UndefinedKey* as its operand

21.2.11 GetValueTest

The **GetValueTest** operator takes a single operand

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642685 565

<*key: DictionaryKey*>

The value returned is determined by the context stack, in the context of interpretation. The Dictionaries on the context stack are examined in order, starting with the top Dictionary on the context stack. If a Dictionary is found which contains the *DictionaryKey key*, the **GetValueTest** operator returns two results

<true: *Boolean*>

<*DictRef: DictionaryReference*>

where *DictRef* is a reference to the Dictionary containing *key*. If *key* is not in any Dictionary on the context stack, **GetValueTest** returns the single result

<false: *Boolean*>

NOTE 119 In either case, the top element of the stack after the execution of GetValueTest is a Boolean. The value of the Boolean is true if key is in any Dictionary on the context stack and is false otherwise.

21.2.12 MakeDictionary

The **MakeDictionary** operator takes a single operand

<*size: Cardinal*>

and returns a single result

<*DictRef: DictionaryReference*>

where *DictRef* is a new empty Dictionary with a maximum capacity of *size* key/value pairs.

NOTE 120 Names bound in Dictionaries may be used as *executable Names* in content, as described in 17.1.3.

21.2.13 MakeString

The **MakeString** operator takes a single operand

<*n: Cardinal*>

and returns a single result

<*S: OctetStringReference*>

where *n* is the number of elements in the newly-created OctetString referenced by S. The **MakeString** operator initializes each element of the new OctetString with the value 0.

21.2.14 MakeVector

The **MakeVector** operator takes a single operand

<*n: Cardinal*>

and returns a single result

<*V: VectorReference*>

where *n* is the number of elements in the newly-created Vector referenced by *V*. The **MakeVector** operator initializes each element of *V* with the object of type *Null*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642686 4T1

21.2.15 MakeandStoreDictionary

The **MakeandStoreDictionary** operator takes $(n \times 2) + 1$ operands, where the first operand on the operand stack is an object of type *Mark*, and the remaining $n \times 2$ operands represent n key/value pairs:

<$value_{n-1}$: *Any*>

<key_{n-1}: *Any*>

<$value_{n-2}$: *Any*>

<key_{n-2}: *Any*>

. .

<$value_0$: *Any*>

<key_0: *Any*>

<*mark: Mark*>

and returns a single result

<*DictRef: DictionaryReference*>

The number of values above the mark ($n \times 2$) is determined, and a Dictionary of capacity n is created. The n key/value pairs on the stack are then removed and stored into the Dictionary. All operands, including the Mark, are removed from the stack by the **MakeandStoreDictionary** operator. If there is no Mark object on the stack, **RaiseError** shall be invoked with *UnmatchedMark* as its operand.

21.2.16 MakeandStoreVector

The **MakeandStoreVector** operator takes a list of $n+1$ operands, where the first operand on the operand stack is an object of type *Mark*:

<$value_{n-1}$: *Any*>

. . .

<$value_1$: *Any*>

<$value_0$: *Any*>

<*mark: Mark*>

and returns a single result

<*V: VectorReference*>

The number of values above the mark (n) is determined, and a Vector of length n is created. The n values on the stack are then removed and stored into the Vector; the topmost value is stored into element $n-1$ of the Vector, and the bottom most value is stored into element 0 of the Vector. All operands, including the Mark, are removed from the stack by the **MakeandStoreVector** operator. If there is no Mark object on the stack, **RaiseError** shall be invoked with *UnmatchedMark* as its operand.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642687 338

21.2.17 PopContextStack

The **PopContextStack** operator takes no operands and returns no results. The effect of interpreting the **PopContextStack** operator is to remove the topmost Dictionary from the context stack. If the context stack is in its initial state, consisting only of the default context stack, **RaiseError** shall be invoked with *ContextStackUnderflow* as its operand.

21.2.18 PushContextStack

The **PushContextStack** operator takes a single operand

<DictRef: DictionaryReference>

and returns no results. The effect of interpreting the *PushContextStack* operator is to push the Dictionary referenced by *DictRef* onto the context stack, thereby making it the first Dictionary searched in the current context of execution. If the implementation-dependent maximum number of *DictionaryReferences* on the context stack would be exceeded by this operation, **RaiseError** shall be invoked with *ContextStackOverflow* as its operand.

21.2.19 Put

The **Put** operator takes three operands

<new: Any>

<key: DictionaryKey>

<x: VectorReference, DictionaryReference, or OctetStringReference>

and returns no results; it replaces a single element of a Vector, Dictionary, or OctetString.

If *x* is a VectorReference or OctetStringReference, the operand *key* shall be of type *Cardinal*, *new* is the new value to be placed at the location indexed by *key*, and the element of the object referenced by *x* whose index is *key* is replaced by *new*. If *key* is greater than or equal to the number of elements in the object referenced by *x*, **RaiseError** shall be invoked with *RangeCheck* as its operand.

If *x* is a DictionaryReference the **Put** operator modifies the Dictionary referenced by *x* as follows: First, the Dictionary is searched for any key/value pair for which the key is equal to *key*. If any such pair is found, it is removed from the Dictionary. Second, the key/value pair (*key*, *new*) is added to the Dictionary. If no key/value pair was found and removed, and if the Dictionary currently has as many pairs as its capacity, the capacity is increased by at least one. If adding this pair would increase the capacity beyond the implementation limit on dictionary size, then **RaiseError** is invoked with *LimitCheck* as its operand.

NOTE 121 Names bound in Dictionaries may be used as *executable Names* in content, as described in 17.1.3.

21.2.20 PutInterval

The **PutInterval** operator takes three operands

<y: VectorReference or OctetStringReference>

<index: Cardinal>

<x: VectorReference or OctetStringReference>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642688 274

and returns no results. *x* and *y* shall be of the same type. The effect is to replace the subsequence of the object referenced by *x* beginning with the element whose index is *index* and extending for the length of *y* elements with the contents of *y*.

If *index* is not a valid index into *x*, or if *x* is not large enough to contain all of *y* beginning at the position indicated by *index*, **RaiseError** shall be invoked with *RangeCheck* as its operand.

21.2.21 PutValue

The **PutValue** operator takes two operands

<*new: Any*>

<*key: DictionaryKey*>

and returns no results. The effect of the **PutValue** operator is determined by the context stack, in the context of interpretation. The Dictionaries on the context stack are examined in order, starting with the top Dictionary on the context stack. The topmost Dictionary which contains the DictionaryKey *key* is altered such that the value to which *key* is bound is replaced by the new value *new*. If *key* is not contained in any of the Dictionaries on the context stack, the **PutValue** operator implicitly executes the **Define** operator to define the key/value pair (*key*, *new*) into the topmost Dictionary on the context stack, increasing the capacity of that Dictionary if necessary.

21.2.22 Search

The **Search** operator takes two operands

<*search: OctetStringReference*>

<*string: OctetStringReference*>

and returns either two or four results, depending on whether or not a substring of the OctetString referenced by *string* matches *search*. If a substring of the OctetString referenced by *string* does match the search string, **Search** returns four results

<true: *Boolean*>

<*pre: OctetStringReference*>

<*match: OctetStringReference*>

<*post: OctetStringReference*>

where *match* is a reference to the first substring of the OctetString referenced by *string* that was matched, *pre* is a reference to the substring of the OctetString referenced by *string* before the matched substring, and *post* is a reference to the unmatched remainder of the OctetString referenced by *string* after the matched substring. If the search string does not match any substring of the OctetString referenced by *string*, **Search** returns two results

<false: *Boolean*>

<*string: OctetStringReference*>

where *string* is a reference to the original string being searched

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642689 100

NOTE 122 In either case, the top element of the stack after the execution of **Search** is a Boolean. The value of the Boolean is true if the search string matches a substring of the string being searched, and false otherwise.

21.2.23 StoreVector

The **StoreVector** operator takes $n+1$ operands, where n is the length of the Vector referenced by the operand VectorReference *V*:

<V: VectorReference>

<value$_{n-1}$: Any>

. . .

<value$_1$: Any>

<value$_0$: Any>

and returns a single result

<V: VectorReference>

The VectorReference *V* is popped from the stack, and its size (*n*) determined. The topmost *n* values on the stack are then removed and stored into the Vector referenced by *V*; the topmost value is stored into element *n-1* of the Vector referenced by *V*, and the bottom most value is stored into element *0* of the Vector referenced by *V*. *V* is then pushed back onto the stack. If the stack does not have at least n values, then **RaiseError** shall be invoked with *StackUnderflow* as its operand.

21.2.24 VectorLoad

The **VectorLoad** operator takes a single operand

<V: VectorReference>

and returns $n+1$ results, where n is the length of the Vector *V*:

<V: VectorReference>

<value$_{n-1}$: Any>

. . .

<value$_1$: Any>

<value$_0$: Any>

The VectorReference *V* is popped from the stack, then the elements of the Vector referenced by *V* are successively pushed onto the operand stack, beginning with the element whose index is 0, and finally *V* is pushed back onto the stack.

21.3 Attribute and Resource Operators

21.3.1 CheckIfExecutable

The **CheckIfExecutable** operator takes one operand

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642690 922

<x: VectorReference or Identifier>

and returns one result

<ex: Boolean>

where *ex* is false if the executability attribute of *x* is literal and true if the executability attribute of *x* is executable. See 18.4.1 for a discussion of the executability attribute.

21.3.2 CheckIfReadable

The **CheckIfReadable** operator takes one operand

<x: DictionaryReference, OctetStringReference, or VectorReference>

and returns one result

<r: Boolean>

where *r* has value true if the access attribute of *x* is *ReadWrite* or *ReadOnly*., and false if the access attribute of x is *ExecuteOnly* or *NoAccess*. See 18.4.2 for a discussion of the access attribute.

21.3.3 CheckIfWriteable

The **CheckIfWriteable** operator takes one operand

<x: DictionaryReference, OctetStringReference, or VectorReference>

and returns one result

<w: Boolean>

where *w* has value true if the access attribute of *x* is *ReadWrite*, and false if the access attribute of *x* is *ReadOnly*, *ExecuteOnly* or *NoAccess*. See 18.4.2 for a discussion of the access attribute.

21.3.4 FindResource

The **FindResource** operator takes two operands

<restype: Name>

<ID: Identifier>

and returns one result *restype* specifies the type of resource to be found, and shall be one of the following literal Names:

— ColorSpace

— DataSource

— Filter

— FontIndexMap

— Form

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642691 869

— GlyphIndexMap

— IndexedFont

— Pattern

The returned result depends upon *restype*. The type and nature of the returned result for each resource type is specified in the following subclauses:

21.3.4.1 Color Space

If *restype* is <ColorSpace: *Name*>, **FindResource** returns the Color Space Object that is bound to *ID* in the set of Declared Resources. The format and contents of the Color Space Object are as described in clause 34. Within this Color Space Object, the Color Space Family is represented by a value of type *OctetStringReference* with an access attribute of *ReadOnly*. The OctetString contains the canonical character string form of the Public Object Identifier for the Color Space Family Identifier (see 9.6).

NOTE 123 This International Standard requires (see 34.6) that any Color Space Object used in content shall have been obtained using **FindResource**. It is the obligation of the implementation to enforce this requirement. One way this can be done is to record the all the references to Color space Objects that have been returned by **FindResource**, then allow only Color Space Object that have references that are in the set of returned object references to be used as Color Space Objects The OctetStringReferences for the Color Space Family Identifier of the Color Space Object can be treated similarly.

21.3.4.2 Data Source

If *restype* is <DataSource: *Name*>, **FindResource** returns the value of type *StreamObject* that is bound to *ID* in the set of Declared Resources. clause 28 and clause 31 describes the different kinds of Data Sources and the Filter, if any, which is applied to each one by the **FindResource** operator.

21.3.4.3 Filter

If *restype* is <Filter: *Name*>, **FindResource** returns a value of type *OctetStringReference* with an access attribute of ReadOnly that is bound to *ID* in the set of Declared Resources. The OctetString contains the canonical character string form of the Public Object Identifier for the Filter Name. Filter Identifiers are specified in clause 31 and in 9.8.

NOTE 124 This International Standard requires (see 31.3.1) that the Filter Identifier operand of the **Filter** operator shall have a Filter Identifier that has been obtained using **FindResource**. It is the obligation of the implementation to enforce this requirement. One way this can be done is to record the all the *OctetStringReferences* that have been returned by **FindResource**, then allow only OctetStringReferences that are in the set of returned object references to be used as Filter Identifiers.

21.3.4.4 Font Index Map

If *restype* is <FontIndexMap: *Name*>, **FindResource** returns the Font Index Map that is bound to *ID* in the set of Declared Resources. The Font Index Map is an object of type *VectorReference*. The format and contents of the referenced Vector are as described in clause 24 .

21.3.4.5 Form

If *restype* is <Form: *Name*>, **FindResource** returns a value of type *Form Object* that is bound to *ID* in the set of Declared Resources. The Form Object is described in clause clause 33.

NOTE 125 This International Standard requires (see 33.5) that the *Form Object* which is the operand of the **ExecuteForm** operator is obtained via execution of the **FindResource** operator. It is the obligation of the implementation to enforce this

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642692 7T5

requirement. One way this can be done is to record the all the *DictionaryReferences* that have been returned by **FindResource**, then allow only DictionaryReferences that are in the set of returned object references to be used as Form Objects

21.3.4.6 Glyph Index Map

If *restype* is <GlyphIndexMap: *Name*>, **FindResource** returns the Glyph Index Map that is bound to *ID* in the set of Declared Resources. The Glyph Index Map is an object of type *VectorReference*. The format and contents of the referenced Vector are as described in clause 24 .

21.3.4.7 Indexed Font

If *restype* is <IndexedFont: *Name*>, **FindResource** returns a value of type *IndexedFont* that is bound to *ID* in the set of Declared Resources. Indexed Fonts are described in clause 24 .

21.3.4.8 Pattern

If *restype* is <Pattern: *Name*>, **FindResource** returns a value of type *Pattern Object* that is bound to *ID* in the set of Declared Resources. The Pattern Object is described in clause 32.

NOTE 126 This International Standard requires (see 32.5) that the *PatternDictionary* which is the operand of the **MakePattern** operator is obtained via execution of the **FindResource** operator. It is the obligation of the implementation to enforce this requirement One way this can be done is to record the all the DictionaryReferences that have been returned by **FindResource**, then allow only DictionaryReferences that are in the set of returned object references to be used as PatternDictionaries.

21.3.4.9 Otherwise

If the value of *restype* is not one of these Names, **RaiseError** shall be invoked with *UndefinedKey* as its operand. *ID* is the value of an INTERNAL RESOURCE IDENTIFIER that has been bound in structure to a resource via a RESOURCE DECLARATION. If there is no resource of type *restype* which is bound to *ID*, **RaiseError** shall be invoked with *UndefinedResource* as its operand.

21.3.5 QueryResource

The **QueryResource** operator takes two operands

<restype: Name>

<ID: Identifier>

and returns one result.

<known: Boolean>

restype specifies the type of resource to be found, and shall be one of the following literal Names:

— ColorSpace

— DataSource

— Filter

— IndexedFont

— Form

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642693 631

— GlyphIndexMap

— FontIndexMap

— Pattern

If the value of *restype* is not one of these Names, **RaiseError** shall be invoked with *UndefinedKey* as its operand. *ID* is the value of an INTERNAL RESOURCE IDENTIFIER that has been bound in structure to a resource via a RESOURCE DECLARATION. If there is a resource of type *restype* which is bound to *ID*, *known*, the return value of **QueryResource**, shall be true; otherwise *known* shall be false.

NOTE 127 **QueryResource** can be used to determine if a resource is defined and declared under a given Name, thus being able to detect if **FindResource** will do any substitutions for the requested resource. For example, a **FindResource** on an IndexedFont will never fail; some font will be substituted by the implementation.

21.3.6 MakeReadOnly

The **MakeReadOnly** operator takes one operand

<x: DictionaryReference, OctetStringReference, or VectorReference>

and returns one result

<x: DictionaryReference, OctetStringReference, or VectorReference>

MakeReadOnly reduces the access attribute of its operand to *ReadOnly*. If the access attribute of the operand is *ExecuteOnly*, **RaiseError** shall be invoked with *InvalidAccess* as its operand. See 18.4.2 for a discussion of the access attribute.

21.3.7 MakeExecuteOnly

The **MakeExecuteOnly** operator takes one operand

<x: DictionaryReference, OctetStringReference, or VectorReference>

and returns one result

<x: DictionaryReference, OctetStringReference, or VectorReference>

MakeExecuteOnly reduces the access attribute of its operand to *ExecuteOnly*. See 18.4.2 for a discussion of the access attribute.

22 Flow of Control, Procedure, and State Operators

22.1 Execute

The **Execute** operator takes a single operand

<x: Any>

and may return results, depending on the type of its operand as described below:

If *x* is of type *Operator*, it is executed.

If *x* is of type *Procedure*, the effect is to interpret the sequence of tokens in the Procedure as though encountered in content.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642694 578

If *x* is of type *Identifier* and its executability attribute is *executable*, the effect is to interpret it as though it were an executable Name encountered in content .

If *x* is of type *Identifier* and its executability attribute is *literal*, or if *x* is of any other type, it is pushed back onto the operand stack.

22.2 Exit

The **Exit** operator takes no operands and returns no results.

The effect of interpreting the **Exit** operator in a Procedure being interpreted by a **Repeat**, **Loop**, **For**, or **ForAll** looping operator is to terminate the interpretation of that looping operator. In the case of nested looping operators, only the interpretation of the operator interpreting the Procedure containing the **Exit** operator is terminated.

If an **Exit** operator is encountered in any context other than inside a Procedure being interpreted by a **Repeat**, **Loop**, **For**, or **ForAll** operator, **RaiseError** shall be invoked with *InvalidExit* as its operand.

22.3 For

The **For** operator takes four operands

<proc: Procedure>

<limit: Number>

<increment: Number>

<initial: Number>

and returns no results. The Procedure is repeatedly interpreted as though by the **Execute** operator.

Before each interpretation of *proc*, a value is placed on the operand stack as an operand to *proc*; these values run in sequence from *initial* by steps of *increment* to *limit*. Interpretation of the **For** operator ceases when the value to be placed on the operand stack as an operand to *proc* would be greater than *limit* (in the case of a positive *increment*), or when the value would be less than *limit* (in the case of a negative *increment*).

The interpretation of the **For** operator may terminate before *limit* is reached if an **Exit** operator is encountered during an interpretation of *proc*.

22.4 ForAll

The **ForAll** operator takes two operands

<proc: Procedure>

<x: VectorReference, DictionaryReference, or OctetStringReference>

and returns no results. The elements of the object referenced by *x*, or in the case that *x* is a *DictionaryReference*, the key/value pairs of the referenced *Dictionary*, are enumerated, and *proc* is interpreted once for each element (key/value pair) in the object referenced by *x*. The Procedure is repeatedly interpreted as though by the **Execute** operator.

Before each interpretation of *proc*, an element (key/value pair) of *x* is placed on the operand stack as an operand to *proc*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642695 404

If *x* is of type *VectorReference* the elements of the Vector referenced by *x* are pushed onto the operand stack beginning with the element whose index is 0 and continuing sequentially.

If *x* is of type *OctetStringReference* the elements of the referenced OctetString are pushed onto the operand stack beginning with the first octet of the OctetString and continuing sequentially; the individual octets are represented on the operand stack by their Cardinal representations (see 18 2 2).

If *x* is of type *DictionaryReference* the key/value pairs of the referenced Dictionary are pushed onto the operand stack in an arbitrary order; for each key/value pair the key bound by the key/value pair is pushed onto the operand stack first, followed by the value to which the key is bound in the Dictionary.

Interpretation of the **ForAll** operator ceases after *proc* has been interpreted for the last element (key/value pair) in *x*. If there are no elements (key/value pairs) in *x* then *proc* is not executed at all.

The interpretation of the **ForAll** operator may terminate before all elements (key/value pairs) of *x* have been enumerated if an **Exit** operator is encountered during an interpretation of *proc*.

22.5 GetDeviceDescription

The **GetDeviceDescription** operator takes no operands and returns one result

<devdict: DictionaryReference>

where *devdict* is the current value of the *DeviceDescription* state variable (see clause 19).

22.6 If

The **If** operator takes two operands

<proc: Procedure>

<cond: Boolean>

and returns no results; however, if *proc* is interpreted, it may leave results on the operand stack.

If the value of *cond* is true, the Procedure is interpreted as though by the **Execute** operator.

If the value of *cond* is false the Procedure is not interpreted.

22.7 IfElse

The **IfElse** operator takes three operands

<$proc_1$: Procedure>

<$proc_0$: Procedure>

<cond: Boolean>

and returns no results; however, $proc_0$ or $proc_1$ may leave results on the operand stack.

If the value of *cond* is true, $proc_0$ is interpreted as though by the **Execute** operator.

If the value of *cond* is false, $proc_1$ is interpreted as though by the **Execute** operator.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642696 340

22.8 Loop

The **Loop** operator takes a single operand

<proc: Procedure>

and returns no results. The Procedure is interpreted as though by the **Execute** operator repeatedly until an **Exit** operator is encountered during an interpretation of *proc*.

22.9 Noop

The **Noop** operator takes no operands and returns no results. It has no effect on the virtual machine state.

22.10 Repeat

The **Repeat** operator takes two operands

<proc: Procedure>

<n: Cardinal>

and returns no results. The Procedure is interpreted as though by the **Execute** operator *n* times in sequence, or until an **Exit** operator is encountered

22.11 RestoreGraphicsState

The **RestoreGraphicsState** operator takes no operands and returns no results. Interpretation of the **RestoreGraphicsState** operator causes the entire graphics state, including the values of all state variables (see clause 12), to be restored from the last Graphics State saved by interpretation of the **SaveGraphicsState** operator. Interpretation of the **RestoreGraphicsState** operator normally causes a *pop* of the Graphics State Stack. If the saved Graphics State that is on the top of the Graphics State Stack was created implicitly by a **SaveState** operator or at the beginning of the most immediately superior block, no *pop* of the Graphics State Stack is done and the Graphics State Stack is left unchanged.

NOTE 128 only the **RestoreState** operator can remove a Graphics State Stack entry that was created by a **SaveState** operator.

22.12 RestoreGraphicsStateXCP

The **RestoreGraphicsStateXCP** operator takes no operands and returns no results. Interpretation of the **RestoreGraphicsStateXCP** operator causes the entire graphics state, including the values of all state variables except *CurrentPosition* (see clause 12), to be restored from the last Graphics State saved by interpretation of the **SaveGraphicsState** operator.

Interpretation of the **RestoreGraphicsStateXCP** operator normally causes a *pop* of the Graphics State Stack. If the saved Graphics State that is on the top of the Graphics State Stack was created implicitly by a **SaveState** operator or at the beginning of the most immediately superior block, no *pop* of the Graphics State Stack is done and the Graphics State Stack is left unchanged.

22.13 RestoreSavedGraphicsState

The **RestoreSavedGraphicsState** operator takes no operands and returns no results. Interpretation of the **RestoreSavedGraphicsState** operator causes the entire Graphics State, including the values of all state variables (see clause 19), to be restored.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The **RestoreSavedGraphicsState** operator repeatedly pops the Graphics State Stack until it encounters a Graphics State that was saved by the **SaveState** operator or it encounters the Graphics State at the bottom of the Graphics State Stack (see 16.2), then implicitly executes the **RestoreGraphicsState** operator to restore to that Graphics State. The Graphics State that was restored is left on the Graphics State Stack.

22.14 RestoreState

The **RestoreState** operator takes one operand

<s: SaveObject>

and returns no results. The effect of the **RestoreState** operator is to restore the Graphics State, the Graphics State Stack, the State Variables, and the set of Referenced Objects to their state at the time *s* was saved (see 17.1.10).

If the Context Stack or Operand Stack contains ObjectReferences to objects created after *s* was created, **RaiseError** shall be invoked with *InvalidRestore* as its operand.

NOTE 129 The **RestoreState** operator implicitly executes the **RestoreSavedGraphicsState** operator to restore the Graphics State and then pops the Graphics State Stack removing the entry placed there by the **SaveState** operator that created *s*.

22.15 SaveGraphicsState

The **SaveGraphicsState** operator takes no operands and returns no results. Interpretation of the **SaveGraphicsState** operator causes the entire Graphics State, including the values of all State Variables (see clause 19), to be saved on the Graphics State Stack.

22.16 SaveState

The **SaveState** operator takes no operands and returns a single result

<s: SaveObject>

The effect of the **SaveState** operator is to save the current state of the Graphics State on the Graphics State Stack and create a SaveObject *s* that contains the Graphics State Stack, the State Variables, and the set of Referenced Objects (see 17.1.10).

NOTE 130 The **SaveState** operator saves the Graphics State by logically executing the **SaveGraphicsState** operator.

23 Coordinate Transformation Operators

This clause defines the operators that manipulate Transformations, and which affect the *CurrentTransformation* Graphics State Variable. The type *Transformation* and the coefficients associated with Transformations are defined in 18.3.4 .

The definition of the Transformation operators requires the following definitions. If T_1 and T_2 are two Transformations, the mapping which consists of applying the mapping represented by T_1 to the result of the mapping represented by T_2 is also of the form specified in 18.3.4. The Transformation which represents this mapping is referred to as the concatenation of T_1 with T_2 (denoted $T_1 \times T_2$).

NOTE 131 If T_1 and T_2 are two Transformations whose associated matrices are respectively

$$M_1 = \begin{pmatrix} a_1 & b_1 & 0 \\ c_1 & d_1 & 0 \\ e_1 & f_1 & 1 \end{pmatrix}$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642698 113

$$M_2 = \begin{pmatrix} a_2 & b_2 & 0 \\ c_2 & d_2 & 0 \\ e_2 & f_2 & 1 \end{pmatrix}$$

then $T_1 \times T_2$ is the Transformation whose associated matrix is the (usual matrix) product $M_1 \times M_2$, which is defined by

$$M_1 \times M_2 = \begin{pmatrix} (a_1 \times a_2 + b_1 \times c_2) & (a_1 \times b_2 + b_1 \times d_2) & 0 \\ (c_1 \times a_2 + d_1 \times c_2) & (c_1 \times b_2 + d_1 \times d_2) & 0 \\ (e_1 \times a_2 + f_1 \times c_2 + e_2) & (e_1 \times b_2 + f_1 \times d_2 + f_2) & 1 \end{pmatrix}$$

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and exception handling are defined in clause 36. In addition to these operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6.2.

23.1 ScaleT

The **ScaleT** operator takes two operands

<s_2: Number>

<s_1: Number>

and returns a single result

<T: Transformation>

where *T* is the Transformation whose matrix is:

$$\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

NOTE 132 When applied, the effect of this Transformation is to scale the User Coordinate System by a factor of s_1 in the x-axis and s_2 in the y-axis.

23.2 Scale

The **Scale** operator takes two operands

<s_2: Number>

<s_1: Number>

and returns no results. The effect of the **Scale** operator is the same as

{s1 s2 **ScaleT Concat**}

23.3 TranslateT

The **TranslateT** operator takes two operands

<y. Number>

<x: Number>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

and returns a single result

<T: Transformation>

where *T* is the Transformation whose matrix is: $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x & y & 1 \end{pmatrix}$

NOTE 133 When applied, the effect of this Transformation is to translate the coordinate system so that the origin is the point whose coordinates are (x,y) in the current User Coordinate System.

23.4 Translate

The **Translate** operator takes two operands

<y: Number>

<x: Number>

and returns no results. The effect of the **Translate** operator is the same as

{x y **TranslateT Concat**}

23.5 RotateT

The **RotateT** operator takes one operand

<a: Number>

where *a* is in degrees and returns a single result

<T: Transformation>

where *T* is the Transformation whose matrix is: $\begin{pmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{pmatrix}$

NOTE 134 When applied, the effect of this Transformation is to rotate the User Coordinate System *a* degrees in a counter-clockwise direction about the current origin.

23.6 Rotate

The **Rotate** operator takes one operand

<a: Number>

where *a* is in degrees and returns no results. The effect of the **Rotate** operator is the same as

{*a* **RotateT Concat**}

23.7 ConcatT

The **ConcatT** operator takes two operands

<T_2: Transformation>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642700 6T1

<T_1: *Transformation*>

and returns a single result

<T_3: *Transformation*>

where $T_3 = T_1 \times T_2$.

23.8 Concat

The **Concat** operator takes a single operand

<*T: Transformation*>

and returns no results. If T_0 is the Transformation (see 18.3.4) that represents the *CurrentTransformation* at the time at which **Concat** is executed, then **Concat** changes its value to $T \times T_0$.

23.9 SetTrans

The **SetTrans** operator takes a single operand

<*T: Transformation*>

and returns no results. The effect is to set the *CurrentTransformation* Graphics State Variable to $T \times T_i$ where T_i is the Initial *CurrentTransformation* component of the Virtual Machine (see 17.1 15).

23.10 GetTrans

The **GetTrans** operator takes no operands and returns a single result

<*T: Transformation*>

where T is the Transformation for which {T **SetTrans**} would set the *CurrentTransformation* Graphics State Variable to its current value.

24 Character Text and the Indexed Font Architecture

The process of imaging character text is intimately bound to the font architecture. The font architecture used in this International Standard is defined in ISO/IEC 9541, Information processing: Font information interchange. The model for character text and Indexed Fonts is established in this Clause. The Indexed Font architecture defined herein augments the ISO/IEC 9541 font architecture to optimize the presentation of text as specified in this International Standard.

The architecture for Indexed Fonts and character text is comprised of several models. The ISO/IEC 9541 font model and Font Resources are described in 24.1; the model for glyphs and glyph sequences is described in 24.2. The Indexed Font model, which is described in 24.2.4 , augments a Font Resource with a method for glyph identification These models are abstract models used to define the semantics of the text and Indexed Font operators. Except where explicit, visible data structures are specified, the implementation need only mimic the behavior of these models; it need not implement them exactly The precise effect produced if a font whose attributes and structure are specified in ISO/IEC 9541 is used with an SPDL document for the imaging of character text is defined by the data structures and operators defined in clause 25 , clause 26, and clause 27 .

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642701 538

24.1 Fonts

The font model used in this International Standard is that established in ISO/IEC 9541, Information processing: Font information interchange.

An (abstract) font is a collection of shapes having the same basic design, for example,. Garamond Italic, together with information about these shapes both individually and collectively. These shapes represent identified graphical symbols, called glyphs, which may be used for the presentation of information in human readable form. The *glyph* is an abstract shape which is independent of any of the particular shapes used to represent it. A *glyph representation* combines a shape representation with metric attributes, such as positioning information. The shape representation is expressed in some description language. The presentation of a shape representation places a *glyph image* on the presentation medium. The glyph image is a visible instance of the glyph.

24.1.1 Font Resources

Using the terminology of ISO/IEC 9541, a font is represented by a Font Resource. The effects of the presentation of character text using a particular font is specified in terms of a Font Resource which represents the font.

A Font Resource consists of a collection of glyph representations together with descriptive information and font metrics which are relevant to the collection of glyph representations as a whole. The glyph representations consist of shape information together with attributes associated with each of the individual glyphs. Each *glyph representation* consists of a *glyph shape*, which is a description of a shape which represents the glyph, and *glyph metrics*, which specify the dimensions and positioning of the glyph shape. Glyph representations in a Font Resource are identified by *glyph identifiers*. In ISO/IEC 9541, glyph identifiers are Structured Names (see 6.16.5). These Structured Names are converted to values of type *Identifier* before the Content Processor is called (see 10 2).

NOTE 135 Although fonts used in existing implementations are in general not defined according to ISO/IEC 9541, the attributes of these fonts which are used in the presentation of character text are basically equivalent to attributes specified in ISO/IEC 9541. It would therefore be possible to produce font resources whose structure and attributes are as defined in ISO/IEC 9541 which would produce the same results when used for the presentation of character text. The presentation of character text in documents using fonts other than those defined according to ISO/IEC 9541 is to be understood in terms of logically equivalent fonts defined by Font Resources whose structure and attributes do conform to ISO/IEC 9541.

24.1.2 Font Size, scaling and rotation

The glyph attributes which are used in the presentation of character text are defined in a Glyph Coordinate System whose units are proportional to the nominal body size of the font. In addition, glyph shapes are graphical objects, either geometric or raster. They are, therefore, subject to all of the graphical manipulations provided by the application of coordinate transformations. For example, a glyph shape, presented using a particular font, may be scaled over a range of sizes, and may be rotated for use in portrait mode, landscape mode, or any other orientation with respect to the medium.

Since presented text is simply the sequence of glyph images that result from presenting a sequence of glyph shapes, each of which can be transformed, an entire strings of text can be scaled and rotated by altering the *CurrentTransformation* in effect at the time that it is imaged.

24.1.3 Bold and italic text

A particular Font Resource has only a single set of (scalable, rotatable) glyph shapes. The glyph shapes for the presentation of bold or italic character text are obtained by the selection of distinct Font Resources which contain these glyph shapes.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642702 474

24.1.4 Writing modes

The presentation of multi-national text often requires that the glyph shapes represented in a Font Resource be usable in a variety of writing modes. These writing modes include left-to-right writing, top-to-bottom writing, and right-to-left writing. A single Font Resource (used in a single size and orientation) may be used to present text in different writing modes.

Although there is only one set of glyph shapes represented in a Font Resource, there may be multiple sets of the glyph metric attributes, one set for each of several writing modes. The writing mode dependent attributes which are of significance to this International Standard are the positioning point coordinates (ISO9541/PX and ISO9541/PY) and the escapement point coordinates (ISO9541/EX and ISO9541/EY).

NOTE 136 Although the coordinates of the bounding box for the glyph shape, ISO9541/MAXFONTEXT, are also repeated for each writing mode, the coordinate values must always be the same because they are given minimum and maximum extents of the shape within the glyph coordinate system and this does not vary with writing mode.

24.2 Glyphs, Glyph Mappings and Indexed Fonts

24.2.1 Glyph identifiers

The combination of a Font Resource and a glyph identifier uniquely specify a glyph representation. When this glyph representation is imaged the result is a glyph image that is added to the page image.

The *choice of the glyph representations* which are to be used to present an element of character text is the responsibility of the Composition and Layout Process. The presentation of each glyph image is completely defined by selecting a font and using the glyph identifier that corresponds to the glyph representation to select the glyph shape and glyph metrics within that font. This International Standard provides several ways to represent the choices of glyph representations that have been made by a Composition and Layout Process. One representation allows the glyph identifier for a chosen glyph representation to be specified directly. Another representation is optimized for representing a sequence of choices of glyph representations.

24.2.2 Strings

In the presentation of character text, it is common to have a sequence of glyphs, all from the same font, to be presented, one after another. This International Standard provides the *GlyphString* type, which is a representation for a sequence of one or more choices of glyph representations.

GlyphStrings are OctetStringReferences. The sequence of octets in the referenced OctetString is used to derive a sequence of Cardinals. These Cardinals are used to identify the glyph representations to be presented. The algorithms for deriving the Cardinals and for using them to identify glyph representations are specified in 25.6 and 26.4. The text of this International Standard will refer to ""the octets in a GlyphString"" with the understanding that this means the octets in the OctetString referenced by the GlyphString.

24.2.3 Indices

24.2.3.1 Glyph indices

Glyph indices are Cardinals which are used to determine a glyph identifier directly (see 25.6) and thereby access a glyph shape.

24.2.3.2 Composite Fonts and Font indices

In languages with ideographic glyphs, the collection of glyphs may be quite large. This International Standard provides for combining a collection of smaller fonts into a hierarchical structure called a Composite Font (see 24.3.1.2). When the current font is such a Composite Font, then some of the Cardinals provided by GlyphStrings

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

are associated with, and specify, component fonts of the Composite Font. These Cardinals are called *font indices*. Fonts without any hierarchical structure are called Base Fonts (see clause 24).

24.2.4 Indexed Fonts

In this International Standard, the ISO/IEC 9541 concept of Font Resource is extended. An *Indexed Font* is a combination of a Font Resource and a Glyph Mapping. An IndexedFont is a value of type *IndexedFont* (see 18.3.1).

24.2.5 Glyph Mappings

A Glyph Mapping is a mapping of Cardinals which may be glyph or font indices to glyph identifiers in the case of glyph indices and to Cardinals, which are used as font selectors, in the case of font indices. This mapping is comprised of both a data structure, the Encoding Vector, and an algorithm. This mapping is specified in Structure as a GLYPH INDEX MAP SPECIFICATION if the result is a Base Font and as part of a COMPOSITE FONT SPECIFICATION if the result is a Composite Font.

The mapping algorithm specifies the algorithm for interpreting the octets in a GlyphString as glyph indices if the font is a Base Font, and glyph and font indices if the font is a Composite Font. The Encoding Vector for a Base Font specifies the association of glyph identifiers with the glyph indices. In the case of a Composite Font, the Encoding Vector specifies the association of component IndexedFonts with font indices. GLYPH INDEX MAPS are represented by Glyph Index Encoding Vectors and FONT INDEX MAPS are represented in Content as Font Index Encoding Vectors.

Specific Glyph Mappings and their application to GlyphStrings are defined in clause 25 and clause 26.

NOTES

137 When a GlyphStrings is imaged, it is in the context of a particular font with a particular Glyph Mapping. The font and mapping used to image the GlyphString is established by the value of the Graphics State Variable *CurrentFont* when the imaging operator which takes the OctetStringReference as an operand is interpreted The method by which GlyphStrings are presented is defined in clause 25 , clause 26 and clause 27

138 It is important to emphasize that this International Standard specifies only the effect to be achieved, and not the mechanism which must be used to produce that result. Although the definitions of this clause are based on the identification of the glyph representations in the Font Resource by glyph identifiers and the explicit mapping of glyph indices to glyph identifiers, it is likely that actual implementations will organize the glyph representations and font information internally in such a way that this mapping is optimized. In addition, many of the fonts which will be used in the presentation of SPDL documents will never exist in interchange format, and will have an assumed Glyph Mapping embedded in their internal representation. The use of such fonts is conforming so long as the effects defined by this International Standard are achieved.

24.3 Indexed Fonts

In this International Standard, Indexed Fonts may be specified in Structure and/or constructed in Content. An Indexed Font is specified in Structure and placed in the environment either by a RESOURCE DEFINITION for a FONT SPECIFICATION (see clause 9 and clause 10) or it is placed in the environment by some process that is outside of the scope of this International Standard. Indexed Font Resources are made available to Content Processing via RESOURCE DECLARATIONS (see clause 9) which put resources into the Set of Available Resources (see clause 16 and clause 17. IndexedFonts may be constructed in Content as outlined in the following subclauses and detailed in clause 25 and clause 26 .

24.3.1 Indexed Font Specification Dictionaries

The Indexed Font Specification Dictionary is a data structure which represents the structure and attributes of an IndexedFont. The form of an Indexed Font Specification Dictionary is consistent with the content notation of

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642704 247

this International Standard. The definition of the Indexed Font Specification Dictionary formally links the font attributes defined in ISO/IEC 9541 with this International Standard. It provides a representation of the Font Resource structure and attributes defined in ISO/IEC 9541 using SPDL object types. The semantics and attributes of the Font Resource portion of the Indexed Font Specification Dictionary are expressed in terms defined in ISO/IEC 9541.

Any Font Resource whose structure and attributes conform to ISO/IEC 9541 can be represented by an SPDL Indexed Font Specification Dictionary. The two structures used to describe font attributes in ISO/IEC 9541 are Structured Names and property lists.

Property lists are represented in SPDL by one of two object types:

— an object of type Vector, which represents an ordered list of property values, or

— an object of type Dictionary for which:

  - the keys of the key/value pairs in the Dictionary correspond to the Structured Names which identify the attributes in the ISO/IEC 9541 property list;

  - the values of the key/value pairs are the values of the attributes, which may themselves be DictionaryReferences, VectorReferences, or values such as numbers.

ISO/IEC 9541 Structured Names are represented in SPDL Base Indexed Font Specification Dictionaries in one of three ways:

— Some of the Structured Names used in ISO/IEC 9541 are abbreviated or modified within SPDL Indexed Font Specification Dictionaries, but their semantics are preserved. These are represented in SPDL by the *Name* data type

— other Structured Names are represented in SPDL by the *Identifier* data type. Such Identifiers are equivalent to the result of applying the **ConvertToIdentifier** operator to an OctetString which is the canonical string form of the public identifier which is semantically equivalent to the Structured Name as defined in ISO/IEC 9070.

— ISO/IEC 9541 glyph identifiers that are registered according to ISO/IEC 10036 are represented in SPDL Font Dictionaries by Names of the form *afiinnnn* as described in 7.7.4.

NOTE 139 Most of the attributes specified as being in a Base Indexed Font Specification Dictionary are defined in ISO/IEC 9541 or are similar to attributes defined in ISO/IEC 9541. In addition, there are some attributes that are used in Base Indexed Font Specification Dictionaries that are defined by this International Standard; some examples are:

— an initial Transformation, representing the size and orientation of the Glyph Coordinate System relative to the User Coordinate System, with the font at nominal unitary size (prior to any use of **ScaleFont** or **TransformFont**).

— a default writing mode for the font.

— a default Glyph Mapping, associating glyph indices with glyph identifiers (see 24.2.1.1).

This International Standard specifies *required attributes* which must be present and *optional attributes* which may be present in an Indexed Font Specification Dictionary or in dictionaries it references. Other key/value pairs, either for other ISO/IEC 9541 attributes or for any other purpose, may occur in these dictionaries and shall be ignored by the Content Processor when using these dictionaries in imaging activities.

A Indexed Font Specification Dictionary can be constructed by the interpretation of token sequences and used to create a IndexedFont which can then be used within a document for the presentation of character text.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642705 183

An Indexed Font Specification Dictionary is a Dictionary containing the key/value pair <*FontType: Integer*>, and additional required key/value pairs depending on the value associated with *FontType*.

There are two categories of Indexed Font Specification Dictionary: Base Fonts are specified using Base Indexed Font Specification Dictionaries and Composite Fonts are specified using Composite Indexed Font Specification Dictionaries. From this point on, a Base Font will be an IndexedFont derived from a Base Indexed Font Specification Dictionary and a Composite Font will be an IndexedFont derived from a Composite Indexed Font Specification Dictionary.

The definition of the character text operators requires the definition of two specialized data structures which are used for presentation of character text. These are Indexed Font Specification Dictionaries and IndexedFonts. clause 25 defines Base Indexed Font Specification Dictionaries together with the mapping of glyph indices. Clause 26 defines Composite Indexed Font Specification Dictionaries together with the related mappings of font and glyph indices. This subclause provides an overview of both kinds of Indexed Font Specification Dictionaries.

24.3.1.1 Base Indexed Font Specification Dictionaries

The Base Indexed Font Specification Dictionary is the data structure which represents IndexedFonts that have no Hierarchical structure. It is the data structure which formally defines the properties of a Base Font. Table 30 defines the following allowed values for *FontType* for Base Indexed Font Specification Dictionaries. The semantics for each type of Base Indexed Font Specification Dictionary is specified in clause 25.

Table 30

Type	meaning
3	An Indexed Font Specification Dictionary with *FontType* 3 can be specified entirely in the SPDL content notation. The semantics of *FontType* 3 Indexed Font Specification Dictionaries underlie the semantics of all Base Fonts
1	An Indexed Font Specification Dictionary with *FontType* 1 corresponds to a Base IndexedFont created using the Indexed Font Interchange Format defined in annex c. The semantics of *FontType* 1 Indexed Font Specification Dictionaries are defined in terms of *FontType* 3 Indexed Font Specification Dictionaries.

24.3.1.2 Composite Indexed Font Specification Dictionaries

Composite Fonts are Hierarchical data structures which provide for the combination of Base Fonts (and other Composite Fonts) into larger, *composite* fonts. Composite Fonts also provide for more complex Glyph Mappings than Base Fonts.

The Composite Indexed Font Specification Dictionary is the data structure which formally defines the properties of a Composite Font (see clause 26). The contents of a Composite Indexed Font Specification Dictionary are similar to those of a Base Indexed Font Specification Dictionary, but glyph shape information is specified via component fonts. Table 31 defines the following allowed values for *FontType* for Composite Indexed Font Specification Dictionaries. The semantics for each type of Composite Indexed Font Specification Dictionary is specified in clause 26.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 31

Type	meaning
0	An Indexed Font Specification Dictionary with *FontType* 0 corresponds to a Composite IndexedFont. The semantics of *FontType* 0 Indexed Font Specification Dictionaries define the semantics of all Composite Fonts.

24.3.2 Indexed Fonts

An IndexedFont is a value of type *DictionaryReference*, which refers to a Dictionary containing the information necessary for the actual process of presenting character text using a particular font. The IndexedFont may be an abstraction of the Indexed Font Specification Dictionary to which it corresponds. An Indexed Font Dictionary shall contain at least:

- glyph shape information;
- as many sets of glyph metrics (to be used in sequentially positioning glyph shapes from the Font Resource) as are needed to support the writing modes used;
- a Transformation representing the relationship of the Glyph Coordinate System to the User Coordinate System, at the size and orientation at which the Font Resource is to be used;
- the writing mode associated with the IndexedFont,
- a Glyph Mapping association glyph indices with glyph identifiers and, in the case of a Composite Font, associating font indices with component IndexedFonts;

IndexedFonts may be constructed by the **DefineFont** operator or selected from the repertoire available in the environment of the Presentation Process by the **FindFont** operator, or produced by operators which modify existing IndexedFonts.

NOTES

140 There is a Font Installation Process outside of the scope of SPDL which accepts a Font Resource as defined for interchange by ISO/IEC 9541 and produces a data structure which can be used to construct a IndexedFont for use by SPDL processors. The function performed by this process is similar to that performed by the **DefineFont** operator, the major differences being that the Font Installation Process binds only a default Transformation and Glyph Mapping to the installed font, and that the Font Installation Process must retain sufficient attribute information associated with the installed font to enable the Presentation Process to select the installed font which best satisfied the font reference which is declared in the FONT REFERENCE FONT SPECIFICATION (see clause 10)

141 The Glyph Mapping of a Base Font may be modified within the document in structure in the REMAPPED FONT SPECIFICATION, or in content by use of the **OpenFont** operator, the **DefineFont** operator, and operators that manipulate Dictionaries and Vectors. The Glyph Mapping of a Composite IndexedFont may not be modified in structure, except by the Font Installation Process; however, a Composite Font mapping may be generated in structure in a COMPOSITE FONT SPECIFICATION. In content, it can be manipulated using the corresponding Composite Indexed Font Specification Dictionary (if it is not closed) and the operators used with Base Fonts immediately above.

142 The definition of IndexedFonts and Indexed Font Specification Dictionaries in this clause and the next two clauses defines their logical structure and provides a basis for the definition, in a subsequent clause, of the effects of using fonts for the imaging of character text. There is no requirement that implementations actually store and access IndexedFonts by means of their associated Indexed Font Dictionaries; it is the intent of the International Standard to enable the effective application of optimization techniques in this area (hence, the provision of *closed* IndexedFont Dictionaries; see 24.4). All that is required is that the effect of interpreting the character text and font operators be as defined.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642707 T56

24.4 Indexed Font Dictionaries

An Indexed Font Dictionary is a Dictionary which has its semantics defined entirely by reference to an Indexed Font Specification Dictionary. The value of the Graphics State Variable *CurrentFont* must be a DictionaryReference referencing a Indexed Font Dictionary. Thus, Indexed Font Dictionaries are the data structure actually used by the interpreter to image glyph representations on the presentation medium.

A Indexed Font Dictionary is said to correspond to a Indexed Font Specification Dictionary containing equivalent metric, glyph shape, and Glyph Mapping information. The effect of using a IndexedFont in the presentation of character text is defined by the operators in this clause with reference to the Indexed Font Specification Dictionary to which it corresponds.

NOTE 143 This clause specifies only the results of using a properly constructed IndexedFont with an SPDL interpreter. The actual method by which these results are achieved by such an interpreter is implementation dependent.

Indexed Font Dictionaries have the following properties:

— Each Indexed Font Dictionary which has an access attribute of ReadOnly corresponds to a Indexed Font Specification Dictionary. When the **DefineFont** operator is applied to a IndexedFont. When the **OpenFont** operator is applied to a IndexedFont, the result is a corresponding Indexed Font Specification Dictionary Reference (unless the referenced Indexed Font Dictionary's access attribute is *ExecuteOnly* or *NoAccess*, see below).

— All Indexed Font Dictionaries have an access attribute of at most ReadOnly. They may not be modified, but there are operators in clause 27 that create a new Indexed Font which is a modification of an existing Indexed Font. In particular, the **Put** operator may not be used with Indexed Font Dictionaries. Such an attempt will result in an *InvalidAccess* exception.

— Implementations may impose the following additional restriction: some or all Indexed Font Dictionary referenced by IndexedFonts may have an access attribute of *ExecuteOnly* or *NoAccess*, then it must contain sufficient information to reconstruct a corresponding Indexed Font Specification Dictionary when the **OpenFont** operator is applied. And if the Indexed Font Dictionary allows applying the **Get** operator, then the Indexed Font Dictionary shall have at least the keys that would be in the Indexed Font Specification Dictionary.

Although the contents of a Indexed Font Dictionary with an access attribute of *ExecuteOnly* or *NoAccess* are unspecified, the semantics of a IndexedFont used for character text presentation is fully specified, in terms of the corresponding Indexed Font Specification Dictionary. Clause 24 to clause 27 of this International Standard defines the semantics of Indexed Font Dictionaries as though they were identical in content and behavior to Indexed Font Specification Dictionaries, with the understanding that implementations may define the contents of Indexed Font Dictionaries in other ways.

NOTE 144 This means that any font technology can be used with the abstract process defined by this International Standard as long as there is a mapping of that technology into model defined by the Indexed Font Specification Dictionary, the Indexed Font Dictionary and the operators on them. Annex C relates this model to the properties of an ISO/IEC 9541 Font Resource.

As part of this method of definition, the representation

{*indexedfont* /*key* **Get**}

is used to represent the action of obtaining the information from the Indexed Font Dictionary referenced by *indexedfont*, corresponding to the information stored as the value of the key *key* in the corresponding Indexed Font Specification Dictionary, even though the Indexed Font Dictionary referenced by *indexedfont* may have an *ExecuteOnly* attribute, and may not have an explicit entry whose key is *key*. Likewise the notation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642708 992

{*indexedfont* **OpenFont**}

is used to represent the action of obtaining a modifiable copy of the Indexed Font Specification Dictionary corresponding to the Indexed Font Dictionary referenced by *indexedfont*, even though the Indexed Font Dictionary referenced by *indexedfont* may have an *ExecuteOnly* attribute, and therefore not be subject to the **OpenFont** operator.

NOTE 145 The above representations are a notational convenience only, since the **Get** and **OpenFont** operators cannot actually be used with some Indexed Font Dictionaries.

25 Base Indexed Font Specification Dictionaries

This clause provides the specifications for the two kinds of Indexed Font Specification Dictionaries: FontType 3 Indexed Font Specification Dictionaries and FontType 1 Indexed Font Specification Dictionaries. This clause also specifies how GlyphStrings are mapped into glyph identifiers using the Encoding Vector that represents a GLYPH INDEX MAP in an Indexed Font. The Standard GLYPH INDEX MAPs, available on all SPDL presentation systems, are described.

25.1 FontType 3 Indexed Font Specification Dictionaries

In addition to the <*FontType: Integer*> pair, A *FontType* 3 Indexed Font Specification Dictionary shall have the following key/value pairs, the required key/value pairs:

25.1.1 FontMatrix

The mandatory key/value pair <*FontMatrix: Transformation*> specifies the initial or default value of the Transformation used when the Indexed Font Specification Dictionary is made into a corresponding IndexedFont.

25.1.2 FontBBox

The mandatory key/value pair <*FontBBox: VectorReference*> references a Vector of four numbers. This represents the value of ISO9541/MAXFONTEXT, in the order (maxfontext-minx, maxfontext-miny, maxfontext-maxx, maxfontext-maxy).

25.1.3 Encoding

The mandatory key/value pair <*Encoding: VectorReference*> references a Vector of Identifiers that represents a GLYPH INDEX MAP This value is used to perform Glyph Mapping during character text imaging (see 25.6).

25.1.4 Metrics

The mandatory key/value pair <*Metrics: DictionaryReference*> is a DictionaryReference which references a Dictionary containing per-glyph information regarding glyph extents, and glyph escapements for the default writing mode of this font, as follows:

— The keys in the referenced Dictionary are glyph identifiers.

— The value of each key/value pair is a reference to a Vector of six numbers in the following order: (*wx, wy, minx, miny, maxx, maxy*).

— The value of <*wx: Number*> is the difference ISO9541/EX - ISO9541/PX

— The value of <*wy: Number*> is the difference ISO9541/EY - ISO9541/PY

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642709 829

— The value of the remaining four numbers are the values of the components of ISO9541/EXT, the extents of the glyph.

— The writing mode dependent positioning point (ISO9541/PX and ISO9541/PY) for the default writing mode shall be (0,0) for all glyphs in a *FontType* 3 font.

25.1.5 ConstructGlyph

The mandatory key/value pair *<ConstructGlyph: Procedure>* references a Procedure responsible for constructing the glyph representation. Its semantics are constrained by its use in **ShowGlyph** (see 27.3.1)

NOTES

146 The recommended definition of the *ConstructGlyph* Procedure, in the context of **ShowGlyph** (see 27.3), and assuming the presence of a *GlyphProcs* Dictionary as described in this subclause is:

{/GlyphProcs **GetValue Exchange Get Execute**}

147 This Procedure would utilize another key/value pair in the Indexed Font Specification Dictionary, *<GlyphProcs: DictionaryReference>*. The value of *GlyphProcs* would reference a Dictionary containing per-glyph glyph representation procedures, as follows:

— The keys in the referenced Dictionary are glyph identifiers.

— The value of each key/value pair is a Procedure which draws a representation of the glyph which has that key as its glyph identifier.

25.2 FontType 3

A *FontType* 3 Indexed Font Specification Dictionary may contain one or more of the following key/value pairs, the optional key/value pairs, with semantics defined by this International Standard:

25.2.1 WMode

The optional key/value pair *<WMode: Cardinal>* specifies the writing mode to be used with the glyph representations in this font, if other than the default. If *WMode* is not present in the font, the value defaults to zero. A value of zero shall indicate the default writing mode. A value of *N*, greater than zero, shall indicate the Nth alternate writing mode, specified by the $(N - 1)^{th}$ element in the *OtherMetrics* Vector (see below). If the value of *WMode* specifies a writing mode not supported by this font, or is greater than the implementation limit for the maximum value of *WMode*, then the content interpreter shall use the default writing mode for text presentation with this font.

25.2.2 OtherMetrics

The optional key/value pair *<OtherMetrics: VectorReference>* references a Vector of DictionaryReferences. Each DictionaryReference shall reference a Dictionary containing per-glyph information regarding positioning points and escapements for an alternate writing mode of this font, as follows:

— The keys in the referenced Dictionary are glyph identifiers. For every key/value pair in this Dictionary, there should be an key/value pair with the same key in the *Metrics* Dictionary.

— The value of each key/value pair is a reference to a Vector of four numbers in the following order: (*wx, wy, px, py*).

- The value of *<wx: Number>* is the difference ISO9541/EX - ISO9541/PX

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642710 540

- The value of <*wy: Number*> is the difference ISO9541/EY - ISO9541/PY
- The values of *px* and *py* are the coordinates, in x and y, of the positioning point (ISO9541/PX, ISO9541/PY) for this glyph in this writing mode.

In addition to the required key/value pairs, and the optional key/value pairs to support alternate writing modes, a Indexed Font Specification Dictionary shall contain any other key/value pairs needed to support the *ConstructGlyph* key/value pair, and may contain any other key/value pairs as desired.

Finally, all Base Indexed Font Specification Dictionaries shall provide a glyph representation whose glyph identifier is <.notdef: *Name*>. This glyph representation shall be imaged whenever a glyph identifier is specified for imaging that is undefined in this font; i.e., whose shape description information is not provided by this font. The glyph representation associated with this special glyph identifier is at the discretion of the font designer.

25.3 FontType 1 Indexed Font Specification Dictionaries

The semantics of *FontType* 1 Font Dictionaries are defined by reference to *FontType* 3 Font Dictionary semantics. A *FontType* 1 Indexed Font Specification Dictionary shall contain all the required key/value pairs specified in normative annex C. Their semantics are as follows:

25.3.1 Key/Value pairs with same semantics as in FontType 3

<*FontMatrix: Transformation*>, <*FontBBox: VectorReference*>, <*Encoding: VectorReference*>: same semantics as for *FontType* 3 Indexed Font Specification Dictionaries.

25.3.2 PaintType

The mandatory key/value pair <*PaintType: Cardinal*> specifies a value that shall be 2 or 0. The semantics of these values is given in table 32.

Table 32

Value	PaintType
0	indicates the glyph representations will be filled (solid) outlines
2	indicates the glyph representations will be stroked (hollow) outlines

25.3.3 CharStrings

The mandatory key/value pair <*CharStrings: DictionaryReference*> references a Dictionary containing per-glyph glyph representation procedures, as follows:

— The keys in the referenced Dictionary are glyph identifiers

— The value of each key/value pair is a reference to an OctetString which is interpreted as an encrypted glyph procedure in the format defined in ISO/IEC 9541-3

25.3.4 FontType 3 keys that are not present in FontType 1

The following key/value pairs which are present in a *FontType* 3 Indexed Font Specification Dictionary are not present in a *FontType* 1 Indexed Font Specification Dictionary. Their semantics are replaced as follows:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— *<Metrics: DictionaryReference>*: The per-glyph information regarding extents and escapement are included in the glyph procedures in the *CharStrings* Dictionary; therefore, looking up the glyph metrics in a *Metrics* Dictionary during glyph imaging is unnecessary. The escapement and positioning information included in the glyph procedures can be overridden by an optional *Metrics* Dictionary (see below), but the key/value pairs in such a Dictionary have different semantics than the key/value pairs in a *Metrics* Dictionary in a *FontType* 3 Indexed Font Specification Dictionary.

— *<ConstructGlyph: Procedure>*: The glyph construction procedure is implicit for a *FontType* 1 Indexed Font Specification Dictionary, and always has the following semantics (refer to 27.3):

- use the glyph identifier to load the corresponding glyph procedure from the *CharStrings* Dictionary in this font;
- interpret the glyph procedure according to the semantics defined in ISO/IEC 9541-3.

25.4 FontType 1 Indexed Font Specification Dictionary

A *FontType* 1 Indexed Font Specification Dictionary may contain one or more of the following key/value pairs, the optional key/value pairs, with semantics defined by this International Standard:

25.4.1 Metrics

The optional key/value pair *<Metrics: DictionaryReference>* references a Dictionary containing per-glyph information. This information can override some of the information included in the glyph procedures, as follows:

— The keys in the referenced Dictionary are glyph identifiers, identical to some or all of the glyph identifiers in the Dictionary referenced by *CharStrings* in the Indexed Font Specification Dictionary.

— The value of each key/value pair may be either

- a single number, *wx*;
- a reference to a Vector of two numbers, (*sbx*,*wx*);
- a reference to a Vector of four numbers, (*sbx*, *sby*, *wx*, *wy*).

— Each glyph procedure in the *CharStrings* Dictionary of a *FontType* 1 Indexed Font Specification Dictionary starts with an *hsbe* or *sbe* glyph procedure operator (see ISO/IEC 9541-3.2 for a description of the semantics of these glyph procedure operators). The numbers *sbx*, *sby*, *wx*, *wy*, override the invocation of the *hsbe* or *sbe* operators as follows:

- The value of *<wx: Number>* is a new escapement (x component) for the default writing mode of the glyph; this is the difference ISO9541/EX - ISO9541/PX.
- The value of *<wy: Number>* is a new escapement (y component) for the default writing mode of the glyph; this is the difference ISO9541/EY - ISO9541/PY. If *wx* is specified, but *wy* is unspecified, then *wy* shall be assigned a value of zero.
- The value of *<sbx: Number>* is a new initial reference point (x component) for the glyph procedure graphic operators. Providing a different value for this component causes the glyph to be displaced horizontally, in all writing modes, by the amount *(override value - glyph procedure value)*. If unspecified, then *sbx* shall be unchanged.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642712 313

- The value of <*sby: Number*> is a new initial reference point (y component) for the glyph procedure graphic operators. Providing, a different value for this component causes the glyph to be displaced vertically, in all writing modes, by the amount *(override value - glyph procedure value)*. If neither *sbx* nor *sby* is specified, then *sby* shall be unchanged. If *sbx* is specified, but *sby* is unspecified, then *sby* shall be assigned a value of zero.

NOTE 148 In fonts which have a default left-to-right writing mode, the value of *sbx* should be the left sidebearing of the glyph (this is the difference ISO9541/EXT/MINX - ISO9541/PX) and the value of *sby* should be zero. See ISO/IEC 9541-3.2, clause 2.8.4 1).

25.4.2 Metrics2

The optional key/value pair < *Metrics2: DictionaryReference*> references a Dictionary containing per-glyph information which, for the first alternate writing mode of this font, can override the default glyph positioning points and glyph escapements included in the glyph procedures. The Dictionary referenced shall be of the same form as those referenced by the *OtherMetrics* key/value pair in a *FontType* 3 Indexed Font Specification Dictionary. If an *OtherMetrics* Vector is also present in this font, the *Metrics2* key/value pair shall take precedence over the zero'th element in the *OtherMetrics* Vector. If a *Metrics* key/value pair is also present in this font, the information in the *Metrics2* Dictionary shall be interpreted with reference to the modified glyph information derived from application of the information in the *Metrics* Dictionary.

25.4.3 OtherMetrics

The optional key/value pair <*OtherMetrics. VectorReference*> shall be as for *FontType* 3 Indexed Font Specification Dictionaries. Information in *OtherMetrics* shall take precedence over corresponding information included in the glyph procedures. In a *FontType* 1 Indexed Font Specification Dictionary, if an *OtherMetrics* key/value pair is present, then a *Metrics2* key/value pair shall be present. If a *Metrics* key/value pair is also present in this font, the information in the *OtherMetrics* Dictionaries shall be interpreted with reference to the modified glyph information derived from application of the information in the *Metrics* Dictionary.

25.4.4 CDevProc

The optional key/value pair <*CDevProc: Procedure*> shall be a reference to a Procedure which is capable of algorithmically modifying per-glyph properties of escapement and positioning point for writing modes 0 and 1. If *Metrics* and/or *Metrics2* Dictionaries are also present in this font, the application of *CDevProc* occurs after the application of information in these Dictionaries. If an *OtherMetrics* key/value pair is present in this font, then *CDevProc* shall not be present. The semantics of *CDevProc* are as follows:

-- Numeric metric information for the first two writing modes (0 and 1) are placed on the operand stack in the order wx_0, wy_0, min_x, min_y, max_x, max_y, wx_1, wy_1, px_1, py_1:

  - wx_0, wy_0, wx_1, wy_1, are the escapements (ISO9541/EX - ISO9541/PX, etc.) for writing modes 0 and 1, respectively.

  - min_x, min_y, max_x, max_y, respectively, are the values of the components of ISO9541/EXT, the extents of the glyph.

  - px_1, py_1, are the x and y components of the positioning point (ISO9541/PX, ISO9541/PY) for writing mode 1. (The positioning point for the default writing mode is (0,0), and cannot be changed by use of *CDevProc*.)

-- The glyph identifier of the glyph being imaged is then placed on the operand stack.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642713 25T

— The *CDevProc* Procedure is interpreted. It shall remove the eleven arguments from the stack, and leave as a result ten numbers with semantics similar to the first ten arguments.

— These ten numbers are then used as new values for the corresponding metric properties for the glyph.

25.4.5 WMode

The optional key/value pair <*WMode: Cardinal*> shall be as for *FontType* 3 Indexed Font Specification Dictionaries.

25.5 FontType 1 Indexed Font Specification Dictionaries

As with *FontType* 3 Indexed Font Specification Dictionaries, *FontType* 1 Indexed Font Specification Dictionaries shall contain any other key/value pairs needed to support the imaging of glyph representations, and shall provide a glyph representation whose identifier is <.notdef: *Name*>. The Indexed Font Specification Dictionary may contain any other entries desired.

25.6 Base Font Glyph Mapping

If the current font is a Base Font, this subclause defines the algorithm for performing Glyph Mapping during character text imaging.

Glyph indices are derived from GlyphStrings as follows:

— Each octet in the GlyphString is interpreted as an Cardinal between 0 and 255, inclusive.

— Each resulting Cardinal is utilized as a glyph index

No use is made of font indices, and none are derived from the GlyphString.

The association between glyph indices and glyph identifiers in a Base Font is specified by the *Encoding* Vector. The *Encoding* is a reference to a Vector of Identifiers, and is a required key/value pair in all Base Indexed Font Specification Dictionaries. This Vector, the Glyph Index Encoding Vector, represents a GLYPH INDEX MAP in content. Each Identifier in the Glyph Index Encoding Vector shall be a glyph identifier in the form specified in 7.7.4 . Each glyph index is used as an index into the Glyph Index Encoding Vector, thereby selecting a glyph identifier. As each glyph specifier is determined, it is returned for further processing to the imaging operator in 27.3 that required Glyph Mapping.

NOTE 149 Glyph Index Encoding Vectors can represent more Glyph Mappings than can be specified by a GLYPH INDEX MAP. The GLYPH INDEX MAP is constrained to have Glyph Identifiers which are either Local Names or Structure Names. On the other hand, any Identifier can be used as a glyph identifier in Glyph Index Encoding Vectors constructed in content.

If a glyph index is greater than the size (less one) of the GLYPH INDEX MAP, then the content interpreter shall raise a *RangeCheck* exception, when it attempts to associate the glyph index with a glyph identifier.

Each glyph identifier in the Glyph Index Encoding Vector should be the identifier of a glyph representation in the font. If a glyph identifier is selected from the Glyph Index Encoding Vector, and there is no such glyph representation in the font, then the glyph identifier <.notdef: *Name*> shall be substituted.

NOTE 150 A glyph identifier used in a Indexed Font Specification Dictionary and/or in a Glyph Index Encoding Vector can be any Identifier. Thus, glyph identifiers used in the Glyph Index Encoding Vector in a given Indexed Font Specification Dictionary should be appropriate to the repertoire of glyph identifiers in the font resource within that Indexed Font Specification Dictionary. In particular, the Identifiers used in the Glyph Index Encoding Vector should be the same as the Identifiers used to identify the glyph representations in the Indexed Font Specification Dictionary.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642714 196

25.7 Standard GLYPH INDEX MAPs

This International Standard defines standard GLYPH INDEX MAPs, some of which shall be available on all SPDL presentation systems, as indicated in the following subclauses. Other GLYPH INDEX MAPs may be available on specific presentation systems, or used in specific IndexedFont and Indexed Font Specification Dictionaries. Such systems or IndexedFonts need not use the standard GLYPH INDEX MAPs to be conforming.

The standard GLYPH INDEX MAPs may be declared in RESOURCE DECLARATIONS (see clause 7) and then used in document content within the scope of the DECLARATION.

25.7.1 Latin1 Publishing GLYPH INDEX MAP

A GLYPH INDEX MAP named *GlyphIndexMap::Latin1Publishing* shall be available on all SPDL presentation systems. It is a Vector of 256 Names. Each Name is of the form "afiinnnn" as specified in 10.3 or is the special name ".notdef".

The specific list of Names in *GlyphIndexMap:.Latin1Publishing* is defined in annex E.

25.7.2 Latin1 Publishing (A) GLYPH INDEX MAP

A GLYPH INDEX MAP named *GlyphIndexMap::Latin1PublishingA* is defined by this International Standard. It is a Vector of 256 Names.

The specific list of Names in *GlyphIndexMap::Latin1PublishingA* is defined in annex E.

25.7.3 Algorithmic AFII GLYPH INDEX MAPs

A family of GLYPH INDEX MAPs named *GlyphIndexMap:.IR:.ddd* shall be available on all SPDL presentation systems. Each *d* is an ISO 646 decimal digit. *ddd* shall be the decimal representation of a number between 0 and 4,294,967,040 decimal and *ddd* shall be evenly divisible by 256..

Each member of this family GLYPH INDEX MAPs shall be a Vector of 256 Names. Each Name is of the form *afiinnn* as specified in 10.2. For each specific instance of the family, *GlyphIndexMap::IR::ddd*, each Name, *afiinnn*, in the GLYPH INDEX MAP shall be such that the decimal value of *nnn* is the decimal value of *ddd* plus the index (0 through 255 decimal, inclusive) of the position of that Name within the Vector.

26 Composite Indexed Font Specification Dictionaries

26.1 FontType 0 Indexed Font Specification Dictionaries

In addition to the *<FontType: Integer>* pair, a *FontType* 0 Indexed Font Specification Dictionary shall have at least the following key/value pairs, the required pairs·

26.1.1 FontMatrix

The mandatory key/value pair *<FontMatrix: Transformation>* specifies the initial or default value of the Transformation used when the Indexed Font Specification Dictionary is made into a corresponding IndexedFont. This transformation is effectively concatenated with the *FontMatrix* of each descendant font in the composite font.

26.1.2 Encoding

The mandatory key/value pair *<Encoding: VectorReference>* references a Vector of Cardinals. Each Cardinal is a font selector. This Vector represents a *FONT INDEX MAP* in content This Vector is used to associate font indices with font selectors during Glyph Mapping (see 26.4).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642715 022

26.1.3 FDepVector

The mandatory key/value pair *<FDepVector: VectorReference>* references a Vector of IndexedFonts. Font selectors from the *FontIndexMap* (the *Encoding* key/value pair; see above) are used to select a IndexedFont from the *FDepVector*. This is how a Composite Font accesses component fonts during Glyph Mapping (see 26.4). Component fonts may be Base IndexedFonts or Composite IndexedFonts.

26.1.4 FMapType

The mandatory key/value pair *<FMapType: Integer>* specifies a Glyph Mapping Algorithm. The Integers for the allowed Glyph Mapping Algorithm and the algorithms themselves are specified in 26.4. Other values are reserved for future use of this International Standard.

26.2 Other Key/Value Pairs

Other key/value pairs are required in a *FontType* 0 Indexed Font Specification Dictionary, depending on the value of *FMapType*. These are defined in 26.4 .

26.3 Optional Key/Value Pairs

The optional key/value pairs in a *FontType* 0 Indexed Font Specification Dictionary, with semantics defined by this International Standard, are:

26.3.1 WMode

The optional key/value pairs *<WMode: Cardinal>* is used in the same way that it is used in a Base Font. The value of *WMode* in a Composite Font overrides the value of *WMode* in all component fonts of the Composite Font. If *WMode* is not present in the Composite Font, the value defaults to zero.

In addition to the above key/value pairs, an Indexed Font Specification Dictionary may contain any other key/value pairs desired.

26.4 Composite Font Glyph Mappings

If the current font is a Composite Font, this clause defines the algorithms for performing Glyph Mapping during character text imaging. The following paragraphs define some terms used in the remainder of this clause.

The hierarchical structure of the Composite Font is viewed as a tree. The root node of the tree is the original Composite Font, which is the current font at the beginning of mapping a GlyphString. This font is referred to as the *root font*. The leaf nodes of the tree are Base Fonts, which actually contain glyph shape information. The intermediate nodes (if any) are additional Composite Fonts. Moving in the direction from the root to the leafs is called descending the tree, moving in the direction from the leafs to the root is called ascending the tree. The component fonts of a Composite Font, and their components to any level, are called *descendant fonts*. The font of which a descendant font is a component is called the *parent font*.

NOTE 151 There may be an implementation limit to the depth of nesting of composite fonts.

In general, the octets in a GlyphString are used to derive a sequence of glyph specifiers, where each glyph specifier consists of a pair, (Base Font, glyph identifier). Then each glyph identifier is imaged from its Base Font as though the Base Font were the current font for the duration of that imaging activity.

NOTE 152 Procedures in Base Fonts, e.g., the *ConstructGlyph* Procedure in a *FontType* 3 font or a *CDevProc* Procedure in a *FontType* 1 font, may use the operators **GetSelectedFont** and **GetRootFont** to determine the currently selected Base IndexedFont and the original Composite IndexedFont, respectively.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642716 T69

The association between font indices and *FDepVector* font selectors in a Composite Font is specified by the *Encoding* Vector. The *Encoding* is a reference to a Vector of Cardinals, and is a required key/value pair in all Composite Indexed Font Specification Dictionaries. This Vector, the Font Index Encoding Vector, represents a FONT INDEX MAP. Each font index is used as an index into the Font Index Encoding Vector, thereby selecting a font selector; this font selector is then used as an index into the *FDepVector* to determine the component font to be used for imaging or further component font selection, as described below.

26.4.1 Composite Font Glyph Mapping Algorithms

There are two categories of Composite Font Glyph Mapping algorithm: modal and non-modal. With non-modal mapping algorithms, the mapping algorithm restarts after each glyph specifier is extracted from the GlyphString. With modal mapping algorithms, the algorithm maintains information about a position in the hierarchical tree structure of component fonts Composite Fonts with non-modal mapping algorithms may be descendant fonts of Composite Fonts with modal mapping algorithms, but not vice versa.

As the Composite Font Glyph Mapping algorithm descends the Composite font tree, the behavior of the algorithm depends on the value of *FMapType* in the Composite Font under consideration. Each descent to a Base Font determines a single glyph specifier and constitutes a cycle of the Composite Font Glyph Mapping algorithm Each cycle may use either or both of the modal and non-modal mapping algorithms Each cycle will also consume some number of octets from the GlyphString As each glyph specifier is determined, it is returned for further processing to the imaging operator in 27 3 that required Glyph Mapping After each cycle, the next cycle begins at the currently selected font. The *currently selected font* is the Indexed Font that was last selected by the modal mapping algorithm or the root font if there is no descendant Indexed Font selected by the modal mapping algorithm. At the beginning of mapping a GlyphString, the currently selected font is the root font. This process continues until all octets in the GlyphString have been exhausted

26.4.1.1 Non-modal Glyph mapping algorithms

The non-modal mapping algorithm is used to construct a glyph specifier when the value of *FMapType of* the Composite Font under consideration is one of the non-modal values. The non-modal mapping algorithm is parameterized with sub-algorithms that specify the behavior of the non-modal mapping algorithm: (1) the *initial sub-algorithm*, used when non-modal mapping is initiated, and (2) the *descendent sub-algorithm*, used for all subsequent non-modal mapping in Indexed Fonts descendent from the initial non-modal font. These sub-algorithms are defined in the 26.4.2 - 26.4.5 below. These sub-algorithms control when and how many octets are read from the GlyphString and determine the potential font and glyph indices used in the algorithm.

There are four states in the state machine for the non-modal mapping algorithm:

1 Based on the value of *FMapType* in the currently selected Composite Font, the appropriate initial sub-algorithm is invoked to establish an initial font index and potential glyph index.

2. The font index is used as an index into the Composite Font's Font Index Encoding Vector, to determine the font selector. The font selector is then used as an index into the *FDepVector* to select a component Indexed Font of the Composite Font.

3 If the selected descendant IndexedFont is a Base Font, then the potential glyph index is an actual glyph index. It is used to index into the Base Font's Glyph Index Encoding Vector (see 25.6), determining the glyph identifier. A single glyph specifier is now determined and this cycle of the Composite Font mapping algorithm is complete.

4. If the selected descendant IndexedFont is itself a Composite Font, then the new value of *FMapType* in this new Composite Font is used to determine which of the descendent sub-algorithms is invoked to determine a new font index and a new potential glyph index. This descendent sub-algorithm utilizes the value

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642717 9T5

of the prior potential glyph index, and may extract further information from the GlyphString, as defined in the subclauses below. The non-modal mapping algorithm continues with state 2 above.

If the non-modal mapping algorithm does not terminate with the above algorithm in state 3, **RaiseError** shall be invoked with *RangeCheck* as its operand.

26.4.1.2 Modal Composite Font mapping algorithms

The modal mapping algorithms are used to construct a glyph specifier when the value of *FMapType of* the Composite Font under consideration is one of the modal values. For modal mapping algorithms, the algorithm for determining glyph specifiers passes through four states:

1. Based on the value of *FMapType* in the currently selected Composite Font, the appropriate modal trigger and an initial font index is established and an octet is extracted from the GlyphString.

2. If the octet is a modal trigger, then the currently selected font may be changed. This process is described in each particular algorithm. If the octet is not a modal trigger, the octet is used as the potential glyph index for the currently selected font.

3. If the currently selected IndexedFont is a Base Font, then the potential glyph index is an actual glyph index. It is used to index into the Base Font's Glyph Index Encoding Vector (see 25.6), determining the glyph identifier. A single glyph specifier is now determined and this cycle of the Composite Font mapping algorithm is complete.

4. If the currently selected IndexedFont is itself a Composite Font, then the new value of *FMapType* in this new Composite Font is used to determine which of the Glyph Mapping algorithms is invoked to determine a new font index and a new potential glyph index. If the value of the new *FMapType* is a non-modal value, the non-modal mapping algorithm is begun in state 4. If the value of the new *FMapType* is a modal value, several actions occur. The font index is set to zero. The font index is then used as an index into the currently selected Composite Font's Font Index Encoding Vector, to determine the font selector. The font selector is used as an index into the *FDepVector* to select a component Indexed Font of the Composite Font as the new currently selected font, another octet is extracted from the GlyphString and processing continues at state 2. immediately above.

If the modal mapping algorithm attempts to extract, an octet from the GlyphString and there is no more octets in the GlyphString, then **RaiseError** shall be invoked with *RangeCheck* as its operand.

The specific algorithms for deriving glyph specifier information from GlyphStrings, depending on the value of *FMapType*, are defined in the subclauses below.

26.4.2 8/8 Mapping

If the value of *FMapType* is 2, the non-modal Glyph Mapping algorithm is used. The sub-algorithms are as follows:

Initial sub-algorithm: two octets are extracted from the GlyphString. The first is interpreted as a font index, between 0 and 255, inclusive. The second is interpreted as a potential glyph index, between 0 and 255, inclusive.

Descendent sub-algorithm, the prior potential glyph index is interpreted as the new font index. One further octet is extracted from the GlyphString; it is interpreted as the new potential glyph index, between 0 and 255, inclusive.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642718 831

26.4.3 1/7 Mapping

If the value of *FMapType* is 4, the non-modal Glyph Mapping algorithm is used. The sub-algorithms are as follows:

Initial sub-algorithm: one octet is extracted from the GlyphString. The most significant bit is interpreted as a font index, either 0 or 1. The remaining 7 bits are interpreted as a potential glyph index, between 0 and 127.

Descendent sub-algorithm: the prior potential glyph index is divided by 128; the result is rounded down to the nearest lesser Cardinal, and interpreted as the new font index. The least significant 7 bits of the prior potential glyph index are interpreted as the new potential glyph index, between 0 and 127.

26.4.4 9/7 Mapping

If the value of *FMapType* is 5, the non-modal Glyph Mapping algorithm is used. The sub-algorithms are as follows:

Initial sub-algorithm, two octets are extracted from the GlyphString. The first octet is multiplied by two, and added to the most significant bit of the second octet. The resulting number is interpreted as a font index, between 0 and 511. The remaining 7 bits of the second octet are interpreted as a potential glyph index, between 0 and 127.

Descendent sub-algorithm, one further octet is extracted from the Glyph String. The prior potential glyph index is multiplied by two, and added to the most significant bit of the further octet. The resulting number is interpreted as the new font index The remaining 7 bits of the further octet are interpreted as the new potential glyph index, between 0 and 127.

26.4.5 Interval Mapping

If the value of *FMapType* is 6, the non-modal Glyph Mapping algorithm is used and there is an additional required key/value pair in the Composite Indexed Font Specification Dictionary: *<SubsVector: OctetString>*. The first octet of the SubsVector determines the *extraction unit size*, *<sz: Cardinal>*. This octet is a Cardinal between 0 and 255 and *sz* is one greater than the value of the first octet.

NOTE 153 There may be an implementation limit on the maximum allowed size of the value *sz*

The remainder of the *SubsVector* OctetString defines a sequence of ranges of consecutive extraction unit values. Each range is defined by *sz* octets; the octets are interpreted as a Cardinal, most significant octet first. The resulting number gives the size of the range of extraction unit values. The sum of all range sizes shall be less than the number 256 to the power of *sz*, which is the total number of extraction unit values expressible in *sz* octets. There is an implicit range at the end of the sequence of explicit ranges, containing the remainder of all possible extraction unit values.

Given an extraction unit value, the Interval mapping algorithm determines a font index and potential glyph index as follows: The initial potential glyph index is set to the extraction unit value and the initial font index is set to zero. For each range in the *SubsVector* OctetString, if the value of the potential glyph index is within that range, the Interval mapping sub-algorithm is done and the current font index and potential glyph index are returned Otherwise, a new potential glyph index is computed by subtracting the size of that range from the value of the old potential glyph index and the font index is incremented by one If there are no more ranges in the *SubsVector* OctetString, then the Interval mapping sub-algorithm is done and the current font index and potential glyph index are returned.

The sub-algorithms for the non-modal Glyph Mapping algorithm are as follows:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642719 778

Initial sub-algorithm: *sz* octets are extracted from the GlyphString; the octets are interpreted as a Cardinal, the first octet being the most significant octet. The resulting number is the extraction unit value. The extraction unit value is converted to a font index and potential glyph index as described in the paragraph above.

Descendent sub-algorithm: the prior potential glyph index is multiplied by 256 to the power of (*sz* minus 1). Next, *sz* minus 1 further octets are extracted from the GlyphString; the octets are interpreted as a Cardinal, the first octet being the most significant octet. Finally, the sum of these two numbers is the new extraction unit value. The extraction unit value is converted to a font index and potential glyph index as described in the paragraph above

26.4.6 Escape Mapping

If the value of *FMapType* is 3, the modal Glyph Mapping algorithm is used and there is an additional optional key/value pair in the Composite Indexed Font Specification Dictionary: *<EscChar: Cardinal>*, whose value shall be between 0 and 255, inclusive. This value is the Escape Code, which is used in the Escape Mapping mapping algorithm. If the *EscChar* key/value pair is absent from the Indexed Font Specification Dictionary, a default value of 255 shall be used. The value of *EscChar* in the *RootFont* IndexedFont shall override *EscChar* values specified in component Composite IndexedFonts.

IndexedFonts with *FMapType* 3 may only be component fonts of other IndexedFonts with *FMapType* 3 or 7.

If the value of *FMapType* is 3, the modal Glyph Mapping algorithm is used with state 2 detailed as follows:

2a. The octet extracted from the GlyphString is interpreted as a Cardinal between 0 and 255, inclusive. If the Cardinal is equal to the Escape Code, then a second octet is extracted from the GlyphString, and interpreted as a Cardinal between 0 and 255, inclusive. The parent font of the currently selected font temporarily becomes the currently selected font. If the second Cardinal is equal to the Escape Code, then the temporary currently selected font becomes the currently selected font, another octet is extracted from the GlyphString and processing continues as for the second octet above. If the second Cardinal is not equal to the Escape Code, then the font index is set to the value of that Cardinal. The font index is then used as an index into the (temporary) currently selected Composite Font's Font Index Encoding Vector, to determine the font selector. The font selector is used as an index into the *FDepVector* to select a component Indexed Font of the Composite Font as the new currently selected font, another octet is extracted from the GlyphString and processing continues at 2a above.

2b. If the first Cardinal is not the Escape code, the octet is used as the potential glyph index for the currently selected font.

26.4.7 Double Escape Mapping

If the value of *FMapType* is 7, the modal Glyph Mapping algorithm is used and there is an additional optional key/value pair in the Composite Indexed Font Specification Dictionary: *<EscChar: Cardinal>*, whose value shall be a Cardinal between 0 and 255, inclusive. This value is the Escape Code, which is used in the Double Escape Mapping mapping algorithm. If the *EscChar* key/value pair is absent from the Indexed Font Specification Dictionary, a default value of 255 shall be used. The value of *EscChar* in the RootFont IndexedFont shall override *EscChar* values specified in any component Composite IndexedFonts.

A IndexedFont with FMapType 7 may only be the root font; it cannot be a component of any other Composite Font.

If the value of *FMapType* is 7, the modal Glyph Mapping algorithm is used with state 2 detailed as follows:

2a. The octet extracted from the GlyphString is interpreted as a Cardinal between 0 and 255, inclusive. If the Cardinal is equal to the Escape Code, then a second octet is extracted from the GlyphString, and interpreted as a Cardinal between 0 and 255, inclusive. The parent font of the currently selected font temporarily becomes

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642720 49T

the currently selected font. If the second Cardinal is equal to the Escape Code, then a third octet is extracted from the GlyphString, and interpreted as an Cardinal between 0 and 255, inclusive. This number, plus 256, is the font index. If the second Cardinal is not equal to the Escape Code, then the font index is set to the value of that Cardinal. The font index is then used as an index into the (temporary) currently selected Composite Font's Font Index Encoding Vector, to determine the font selector. The font selector is used as an index into the *FDepVector* to select a component Indexed Font of the Composite Font as the new currently selected font, another octet is extracted from the GlyphString and processing continues at 2a. above.

2b. If the first Cardinal is not the Escape code, the octet is used as the potential glyph index for the currently selected font

26.4.8 ShiftOut/ShiftIn Mapping

If the value of *FMapType* is 8, the modal Glyph Mapping algorithm is used and there are two additional optional key/value pairs in the Composite Indexed Font Specification Dictionary: *<ShiftOut: Cardinal>*, and *<ShiftIn: Cardinal>*. The value of each shall be a Cardinal between 0 and 255, inclusive. If the *ShiftOut* key/value pair is absent from the Indexed Font Specification Dictionary, a default value of 14 shall be used. If the *ShiftIn* key/value pair is absent from the Indexed Font Specification Dictionary, a default value of 15 shall be used.

A IndexedFont with FMapType 8 may only be the root font; it cannot be a component of any other Composite Font.

If the value of *FMapType* is 8, the modal Glyph Mapping algorithm is used with state 2 detailed as follows:

2a The octet extracted from the GlyphString is interpreted as a Cardinal between 0 and 255, inclusive. If the Cardinal is equal to the ShiftIn code, then the font index is set to zero; if the Cardinal is equal to the ShiftOut code, then the font index is set to 1. The parent font of the currently selected font temporarily becomes the currently selected font. The font index is then used as an index into the currently selected Composite Font's Font Index Encoding Vector, to determine the font selector. The font selector is used as an index into the *FDepVector* to select a component Indexed Font of the Composite Font as the new currently selected font, another octet is extracted from the GlyphString and processing continues at 2a. above

2b. If the Cardinal is neither the ShiftIn code or the ShiftOut code, the octet is used as the potential glyph index for the currently selected font

26.5 Standard FONT INDEX MAPs

This International Standard defines standard FONT INDEX MAPs which shall be available on all SPDL presentation systems. Other FONT INDEX MAPs may be available on specific presentation systems, or used in specific IndexedFont and Indexed Font Specification Dictionaries Such systems or IndexedFonts need not use the standard FONT INDEX MAPs to be conforming

The standard FONT INDEX MAPs may be declared in RESOURCE DECLARATIONS (see clause 7) and then used in document content within the scope of the DECLARATION.

26.5.1 Algorithmic FONT INDEX MAPs

A family of FONT INDEX MAPs named *FontIndexMap/Sequential/nnn* shall be available on all SPDL presentation systems. Each *n* is an ISO 646 decimal digit character. The sequence of digits *nnn* shall represent a decimal number whose value is σ 512.

Each member of this family of FONT INDEX MAPs shall be a Vector of *nnn* Cardinals, where the elements of the Vector shall be the sequential Cardinals 0 through (*nnn* - 1)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642721 326

27 Font and Character Text Operators

27.1 Font Accessing and Manipulation Operators

The following operators are used to obtain, modify, and construct IndexedFonts.

27.1.1 DefineFont

The **DefineFont** operator takes one operand

<FD: DictionaryReference>

and returns a single result

<indexedfont: DictionaryReference>

FD shall be a reference to a valid Indexed Font Specification Dictionary. The returned result *indexedfont* is a reference to a new Indexed Font Dictionary, corresponding to *FD*.

27.1.2 FindFont

The **FindFont** operator takes a single operand

<ID: Name>

where *ID* is the value of an INTERNAL RESOURCE IDENTIFIER which is bound to a IndexedFont in the Context of Interpretation. It returns a single result

<indexedfont: DictionaryReference>

where *indexedfont* is a reference to the Indexed Font Dictionary to which *ID* is bound.

If *ID* is the value of a valid INTERNAL RESOURCE IDENTIFIER in the current context of interpretation and is bound to a *IndexedFont*, the effect of interpreting **FindFont** is the same as the effect of interpreting the **FindResource** operator with a */IndexedFont* operand;. that is, the result of {*ID* **FindFont**} is the same as {*ID* */IndexedFont* **FindResource**}

NOTE 154 The algorithm for selecting the Indexed Font Dictionary which best matches the attributes in a FONT REFERENCE FONT SPECIFICATION is constrained by the SATISFACTION REQUIREMENT and the MATCH RULES and may fail to find a suitable match, but a selection is always provided.

If the font reference is unsatisfied (whether for lack of a definition of a suitable Indexed Font Resource, for lack of a declaration of ID as the name of an Indexed Font Resource, or from failure to *satisfy a font reference* as defined in clause 10) **RaiseWarning** will be invoked with *FailureToSatisfyFontReference* as its operand.

27.1.3 GetRootFont

The **GetRootFont** operator takes no operands and returns one result

<indexedfont: DictionaryReference>

where *indexedfont* is a reference to the Indexed Font Dictionary specified by the current value of the *CurrentFont* imaging variable (set by the most recent use of the **SetFont** operator).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

27.1.4 GetSelectedFont

The **GetSelectedFont** operator takes no operands and returns one result

<indexedfont: DictionaryReference>

where *indexedfont* is a reference to a Indexed Font Dictionary, except during character text imaging, this value shall be the same as that returned by the **GetRootFont** operator. However, if **GetSelectedFont** is executed by a Procedure during character text imaging (e.g., as part of a glyph procedure) while the value of the *CurrentFont* imaging variable is a Composite Font, it shall instead return a reference to the currently selected Base Indexed Font Specification Dictionary. (See 26.4.)

27.1.5 OpenFont

The **OpenFont** operator takes one operand

<indexedfont: DictionaryReference>

and returns one result

<FD: DictionaryReference>

indexedfont shall be a reference to a Indexed Font Dictionary. The returned result *FD* is a reference to a new, modifiable copy of the Indexed Font Specification Dictionary corresponding to *indexedfont*

If the Indexed Font Dictionary referenced by *indexedfont* has an access attribute of *ExecuteOnly* or *NoAccess*, interpreting **OpenFont** shall cause **RaiseError** to be invoked with *InvalidAccess* as its operand.

27.1.6 PutWMode

The **PutWMode** operator takes two operands

<wm: Cardinal>

<indexedfont1: DictionaryReference>

where *wm* is the number of a writing mode, and *indexedfont1* is a reference to a Indexed Font Dictionary. It returns a single result

<indexedfont2: DictionaryReference>

where *indexedfont2* is a reference to the new Indexed Font Dictionary which would be the result of

{*indexedfont1* **OpenFont Dup** *<WMode: Cardinal>* *wm* **Put DefineFont**}

27.1.7 ScaleFont

The **ScaleFont** operator takes two operands

<sc: Number>

<indexedfont1: DictionaryReference>

where *indexedfont1* is a reference to a Indexed Font Dictionary, and returns a single result

<indexedfont2: DictionaryReference>

where *indexedfont2* is a reference to the new Indexed Font Dictionary which would be the result of

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642723 1T9

{*indexedfont1* [*sc* 0 0 *sc* 0 0] **TransformFont**}.

27.1.8 TransformFont

The **TransformFont** operator takes two operands

<T2: Transformation>

<indexedfont1: DictionaryReference>

where *indexedfont1* is a reference to a Indexed Font Dictionary, and returns a single result

<indexedfont2: DictionaryReference>

where *indexedfont2* is a reference to the new Indexed Font Dictionary.

If *indexedfont1* references a Base Font, then *indexedfont2* references a new Base Font which would be the result of

{*indexedfont1* **OpenFont Dup Dup**

/*FontMatrix***Get**

/*FontMatrix***Exch**

T2 **ConcatT Put DefineFont**}

If *indexedfont1* references a Composite Font, then *indexedfont2* references a new Composite Font which, when used for imaging, shall image glyphs as though the above transformation had been applied to each descendant base font.

NOTE 155 In either case, the value of the associated Transformation is not replaced by *T2* but is incrementally modified by concatenation with *T2*.

27.2 Graphics State Manipulation Operators

The following operators are used to set and access the *CurrentFont* Graphics State Variable and the *CurrentPoint* Graphics State Variable.

27.2.1 GetPosition

The **GetPosition** operator takes no operands and returns two results

<y: Number>

<x: Number>

where {*x y* **SetPosition**} would set the value of the Graphics State Variable *CurrentPosition* to its current value.

27.2.2 SetFont

The **SetFont** operator takes a single operand

<indexedfont: DictionaryReference>

where *indexedfont* is a reference to a Indexed Font Dictionary, and returns no results. It sets the value of the Graphics State Variable *CurrentFont* to *indexedfont*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642724 035

27.2.3 SetPosition

The **SetPosition** operator takes two operands

<y: Number>

<x: Number>

and returns no results. It sets the value of the Graphics State Variable *CurrentPosition* to the point whose coordinates are (*x*, *y*) in the current User Coordinate System. It begins a new path segment (see clause 29) with a start point and end point of (*x*, *y*) in UCS coordinates. If the last path segment of the *CurrentPath* Graphics State Variable consisted solely of a single point path element; that path segment is deleted and replaced by the new path segment created by the execution of the **SetPosition** operator.

27.2.4 SetPositionRelative

The **SetPositionRelative** operator takes two operands

<y: Number>

<x: Number>

and returns no results. It changes the value of the Graphics State Variable *CurrentPosition* from the point whose coordinates are (*CPx*, *CPy*) in the current User Coordinate System to the point whose coordinates are (*CPx*+*x*, *CPy*+*y*) It begins a new path segment (see clause 29) with a start point and end point of (*CPx*+*x*, *CPy*+*y*) in UCS coordinates. If the last path segment of the *CurrentPath* Graphics State Variable consisted solely of a single point path element; that path segment is deleted and replaced by the new path segment created by the execution of the **SetPositionRelative** operator

If no *CurrentPosition* is currently defined, **RaiseError** shall be invoked with *NoCurrentPosition* as its operand.

27.3 Character Text Operators

The following operators are used to present character text.

27.3.1 ShowGlyph

The definition of the **ShowGlyph** imaging operator provides the linkage between the font architecture and the imaging of character text in SPDL All other operators which image character text are defined in terms of **ShowGlyph**.

The imaging operator **ShowGlyph** takes a single operand

<glyphid: Identifier>

and returns no results. The effect of interpreting **ShowGlyph** depends on the value of the Graphics State Variable *CurrentFont*.If the value of *CurrentFont* is a Base Font, *indexedfont*, then the effect of {*glyphid* **ShowGlyph**} is as follows. (A line beginning with a percent sign ('%') is a comment and not part of the Procedure which defines **ShowGlyph**)

```
{%First, save the graphics state
   SaveGraphicsState
%Translate UCS so that origin is at CurrentPosition
   GetPosition Translate
%Apply FontMatrix so CurrentTransformation
%maps Glyph Coordinate System to RCS
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642725 T71


```
    GetSelectedFont /FontMatrix Get Concat
  %Determine writing mode of font and put result on Operand Stack
    GetRootFont /WMode Known
      {GetRootFont /WMode Get}
      {0}
    IfElse
  %Do positioning point offset if using alternate writing mode; save initial point
    SaveGraphicsState
    Dup 0 NotEqual
      {Dup 1 Equal
        {GetSelectedFont /Metrics2 Get}
        {Dup GetSelectedFont /OtherMetrics Get
        Exchange 2 Subtract Get}
       IfElse
  %Correct metrics DictionaryReference is now on top of stack
  %Operand Stack has GlyphID WMode MetricsDictRef


        2 Index Get
        Dup 2 Get Negate Exchange 3 Get Negate
        Translate
      }
    If
  %Image the glyph: with Indexed Font Dictionary on the Context Stack
  %and in a new Path which avoids interaction with any existing path
    GetSelectedFont PushContextStack
    NewPath 1 Index
    /ConstructGlyph GetValue Execute
    PopContextStack
    RestoreGraphicsState
  %Get escapement metrics using writing mode on top of Operand Stack
  %to select the correct metrics Dictionary
    Dup 0 Equal
      {Pop GetSelectedFont /Metrics Get }
      {Dup 1 Equal
        {Pop GetSelectedFont /Metrics Get}
        {GetSelectedFont /OtherMetrics Get
        Exchange 2 Subtract Get}
      IfElse}
    IfElse
  %The Operand Stack has: glyphid metricsDictionaryReference
  %The escapement values are the first two elements of the metrics vector
    Exchange Get Dup 0 Get Exchange 1 Get
  %Move CurrentPosition to (Ex, Ey)
    SetPosition
  %Restore the graphic state, without nullifying escapement
    RestoreGraphicsStateXCP
  }
```

If the current font is a Composite Font or is *Null*, **RaiseError** shall be invoked with *InvalidFont* as its operand. If the value of the *CurrentPosition* Graphics State Variable is undefined, **RaiseError** shall be invoked with *NoCurrentPosition* as its operand.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642726 908

27.3.2 ShowString

The imaging operator **ShowString** takes a single operand

<s: GlyphString>

and returns no results. If the OctetString referenced by *s* is mapped by the Glyph Mapping algorithm to the sequence of glyph specifiers ((*bf1*, *gid1*), (*bf2*, *gid2*), . . , (*bfN*, *gidN*)), where each *bfX* is a Base *IndexedFont* and each *gidX* is a glyph identifier, then the effect of

{*s* **ShowString**}

is the same as the execution of

{*bfX* **SetFont** *gidX* **ShowGlyph**}

for each of the glyph specifiers (*bfX*, *gidX*) in sequence

If the current font is a Base *IndexedFont*, then all *bfX* will simply be the current font. If the current font is a Composite *IndexedFont*, then the use of the **SetFont** operator is figurative, as **GetRootFont** shall continue to return the original value of the *CurrentFont* imaging variable. However, use of **GetSelectedFont** within any Base Font *bfX* shall return *bfX*.

If the current font is *Null*, **RaiseError** shall be invoked with *InvalidFont* as its operand. If the value of the *CurrentPosition* Graphics State Variable is undefined, **RaiseError** shall be invoked with *NoCurrentPosition* as its operand. If during the execution of the Glyph Mapping algorithms, either a font index or a glyph index is greater than or equal to the **Capacity** of the Encoding Vector used with that index, or the algorithm fails in any other way, **RaiseError** shall be invoked with *RangeCheck* as its operand.

27.3.3 ShowStringEscapedX

The imaging operator **ShowStringEscapedX** takes two operands

<v: VectorReference>

<s: GlyphString>

where *v* must reference a Vector of Numbers, and have at least as many elements as the number of glyph specifiers derived from the *GlyphString s*. The operator returns no results

If for $X = 0, \ldots, n$

— (*bfX*, *gidX*) is a single glyph specifier derived from the *GlyphString s*;

— *<vx: Number>* is the Xth element of *v*;

then the effect of {*s v* **ShowStringEscapedX**} is the same as the execution of

{**SaveGraphicsState**
bfX **SetFont** *gidX*
ShowGlyph
RestoreGraphicsState
vX 0 **SetPositionRelative**}

for $X = 0, \ldots, n$ in sequence.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642727 844

If the current font is a Base IndexedFont, then all *bfX* will simply be the current font. If the current font is a Composite IndexedFont, then the use of the **SetFont** operator is figurative, as **GetRootFont** shall continue to return the original value of the *CurrentFont* imaging variable. However, use of **GetSelectedFont** within any Base Font *bfX* shall return *bfX*.

ShowStringEscapedX shall invoke **RaiseError** under the same conditions and with the same operands that **ShowString** shall. In addition, if the Vector of Numbers does not have at least as many elements as the number of glyph specifiers derived from the *GlyphString s*, then **RaiseError** shall be invoked with *RangeCheck* as its operand.

27.3.4 ShowStringEscapedY

The imaging operator **ShowStringEscapedY** takes two operands

<v: VectorReference>

<s: GlyphString>

where *v* must reference a Vector of Numbers, and have the same number of elements as the number of glyph specifiers derived from the *GlyphString s* The operator returns no results.

If for $X = 0, \ldots, n$

— (*bfX*, *gidX*) is a single glyph specifier derived from the *GlyphString s*;

— *<vx: Number>* is the Xth element of *v*;

then the effect of {*s v* **ShowStringEscapedY**} is the same as the execution of

{**SaveGraphicsState**

bfX **SetFont** *gidX* **ShowGlyph**

RestoreGraphicsState

0 vX **SetPositionRelative**}

for $X = 0, \ldots, n$ in sequence.

If the current font is a Base IndexedFont, then all *bfX* will simply be the current font. If the current font is a Composite IndexedFont then the use of the **SetFont** operator is figurative, as **GetRootFont** shall continue to return the original value of the *CurrentFont* imaging variable. However, use of **GetSelectedFont** within any Base Font *bfX* shall return *bfX*.

ShowStringEscapedY shall invoke **RaiseError** under the same conditions and with the same operands that **ShowString** shall. In addition, if the Vector of Numbers does not have at least as many elements as the number of glyph specifiers derived from the *GlyphString s*, then **RaiseError** shall be invoked with *RangeCheck* as its operand.

27.3.5 ShowStringEscapedXY

The imaging operator **ShowStringEscapedXY** takes two operands

<v: VectorReference>

<s: GlyphString>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

where *v* must reference a Vector of Numbers, and have exactly twice the number of elements as the number of glyph specifiers derived from the *GlyphString s*. The operator returns no results.

If for $X = 0, \ldots, n$

— (*bfX*, *gidX*) is a single glyph specifier derived from the *GlyphString s*;

— <*v2X: Number*> and <*v2X+1: Number*> are respectively the 2Xth and the 2X+1st elements of *v*,

then the effect of {*s v* **ShowStringEscapedXY**} is the same as the execution of

{**SaveGraphicsState**

bfX **SetFont** *gidX* **ShowGlyph**

RestoreGraphicsState

v2X v2X+1 **SetPositionRelative**}

for $X = 0, \ldots, n$ in sequence.

If the current font is a Base IndexedFont, then all *bfX* will simply be the current font. If the current font is a Composite IndexedFont then the use of the **SetFont** operator is figurative, as **GetRootFont** shall continue to return the original value of the *CurrentFont* imaging variable. However, use of **GetSelectedFont** within any Base Font *bfX* shall return *bfX*

ShowStringEscapedXY shall invoke **RaiseError** under the same conditions and with the same operands that **ShowString** shall. In addition, if the Vector of Numbers does not have at least two times as many elements as the number of glyph specifiers derived from the *GlyphString s*, then **RaiseError** shall be invoked with *RangeCheck* as its operand

27.3.6 StringWidth

The operator **StringWidth** takes a single operand

<*s: GlyphString*>

and returns two Numbers.

<*wy: Number*>

<*wx: Number*>

where, provided the **ShowString** imaging is not actually done, *wx wy* would be the results returned by

```
{
    SaveGraphicsState
% Get current X,Y coordinates
    GetPosition
% Get coordinates after string is shown
    s ShowString GetPosition
% Take difference of Y coordinates
    3 -1 Roll Subtract
% Take difference of X coordinates
    3 1 Roll Exchange Subtract
% Put delta X before delta Y on stack
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642729 617

Exchange
RestoreGraphicsState
}

If the OctetString referenced by *s* is mapped by the Glyph Mapping algorithm to the sequence of glyph specifiers ((*bf1*, *gid1*), (*bf2*, *gid2*), ... , (*bfN*, *gidN*)), where each *bfX* is a Base IndexedFont and each *gidX* is a glyph identifier, then the effect of

{*s* **StringWidth**}

is the same as the execution of

{*bfX* **SetFont** *gidX* **ShowGlyph**}

for each of the glyph specifiers (*bfX*, *gidX*) in sequence.

If the current font is a Base IndexedFont, then all *bfX* will simply be the current font. If the current font is a Composite IndexedFont, then the use of the **SetFont** operator is figurative, as **GetRootFont** shall continue to return the original value of the *CurrentFont* imaging variable. However, use of **GetSelectedFont** within any Base Font *bfX* shall return *bfX*.

If the current font is *Null*, **RaiseError** shall be invoked with *InvalidFont* as its operand. If during the execution of the Glyph Mapping algorithms, either a font index or a glyph index is greater than or equal to the **Capacity** of the Encoding Vector used with that index, or the algorithm fails in any other way, **RaiseError** shall be invoked with *RangeCheck* as its operand.

28 Raster Graphics Operators

28.1 Model for Raster Graphics

Raster graphics refers to the representation of graphical information by means of information about a rectangular array of rectangular elements called *picture elements* or *pels* This information can be used in either of two ways

— to specify a shape, by specifying the pels which constitute the *inside* of the shape

— to specify an image, by specifying a black and white, gray-scale, or color value for each pel.

The first type of raster graphics is called *bitmap raster graphics*. Bitmap raster graphics are provided in SPDL by means of the **MaskBitMap** operator. The second type of raster graphics is called *sampled raster graphics*. Sampled raster graphics are provided by the **ImageRasterElement** operator.

Both types of raster graphics require similar facilities for the acquisition and processing of the raster graphics image data. They differ primarily in that the raster graphics image data used for bitmap raster graphics consists of a Boolean value for each pel which specifies whether or not that pel is inside the shape being defined, whereas the raster graphics image data used for sampled raster graphics may involve more complex values which specify the value and/or color of each pel.

The operators and data types for raster graphics make possible the presentation of large raster graphics elements even on printers that are incapable of storing all of the raster graphics image data required for a raster graphics image element prior to imaging. It also provides flexibility in the format of raster graphics image data, so that the output of many types of scanners, possibly compressed via a variety of data compression algorithms, can be used.

NOTE 156 It is important to note that SPDL specifies the effect to be achieved in the imaging of a raster graphics element, and not the technique used to produce this effect. It is unlikely that real implementations will perform all of the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642730 339

intermediate steps described in the definitions of the raster imaging operators. The purpose of these descriptions is to facilitate the independent binding of the major components of the raster imaging process to a particular instance of such imaging, and to allow the precise description of the required end result

28.1.1 Sampled Raster Graphics

A sampled raster graphics representation of an image consists of information about the color and/or intensity of the image at a large number of sample points from which the image may be reconstructed. The individual units of information are called sample values, and the collection of the sample values for a particular image is called the sample data for that image. Such raster graphics image data can be produced by scanners by taking measurements on an existing picture at each of the points of a sampling grid, or may be generated synthetically by graphics systems.

The process of scanning a picture to produce a sampled image consists of two main steps:

— taking measurements of the color and/or intensity (luminance or lightness) of the picture at each point of the sampling grid and representing these measurements parametrically with reference to a particular color model;

— packing the parametric representations of the measurements into a compact data structure, frequently using compression techniques to compress this data structure further.

The reconstruction of a graphic image from the sampled data requires that these steps be reversed. The model for the reconstruction and imaging of a raster graphics image involves four steps:

NOTE 157 the following description is a conceptual overview; it is not intended as a complete description of the reconstruction; that is given in 28 3 1 below.

— The input to the reconstruction process is a sequence of sample values arranged in a rectangular array. The array of sample values is *Width* samples wide and *Height* samples high. This array is positioned in UCS by the *ImageMatrix* provided with the sample data. Thus positioned, each element of the array becomes a rectangular area in UCS.

- Each element of the array of sample values can be viewed as being a vector of color components suitable for specifying a color in the *CurrentColorSpace*.

NOTE 158 Examples of sample value vectors are a single element vector consisting of a gray level or a three element vector consisting of a set of RGB color components.

— The sample values are interpreted to specify a color for each of the elements of the array of rectangular areas.

— The array of colored rectangular areas is imaged onto the page image, element by element. Each element is painted onto the page image using the color of that element as the ink, the rectangular area the element occupies as the mask, and the *CurrentClippingRegion* as the clipping region.

NOTE 159 When rendering the page image onto an instance of the media, presentation devices frequently increase their apparent gamuts of visible color by approximating colors which they cannot reproduce directly with spatially dispersed patterns of spots of their primary colors. A common use of this technique is the halftone rendering of images on black-and-white devices to produce the appearance of gray-scale colors. It is also common for raster image creation devices to produce image data which already incorporates such methods for color gamut extension. Such a device might for example, produce bi-level raster data for the representation of gray-scale photographic images. The SPDL raster graphics imaging operators allow the use of either technique for the representation of raster graphics images.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

28.1.2 Bitmap Raster Graphics

Bitmap raster graphics image data consists of a Boolean value for each pel which specifies whether or not that pel is inside the shape being defined. The aggregate of the bitmap raster graphics image data defines a shape which is used as a mask for a imaging operator. For each sample in the bitmap raster graphics image data, one of the Boolean values indicates that the sample is in the opaque part of the mask, and the other Boolean value indicates that the sample is in the part of the mask through which ink is deposited.

NOTE 160 Which values correspond to the opaque part of the mask and which value correspond to the part of the mask through which ink is deposited is specified by the *Decode* key in the *image Dictionary* operand to the **MaskBitMap** operator (see 28.2.1.2).

Although bitmap raster graphics image data is logically two-valued, the manipulation of the raster graphics image data to obtain the final values used to define the mask frequently requires the same kinds of operations as are required for sampled raster graphics image data. For this reason SPDL uses the same facilities for the manipulation of bitmap raster graphics image data and sampled raster graphics image data.

28.1.3 Raster Graphics Image Data

Three important characteristics of raster graphics technology are

— wide variety of methods used for representing colors

— the wide variety of methods used for compressing sample values

— the extremely large volume of data required for representing raster graphics images

While the first of these is applicable only to sampled images, the later two are equally applicable to bitmap raster data.

As a result of these characteristics it is not reasonable to require that all systems which deal with raster graphic data translate all raster data into a single standard representation in order to be able to use the SPDL syntax and semantics. SPDL defines a syntax for raster graphics image processing which allows for the use of non-standard as well as standard color representation methods and data compression methods. SPDL defines standard ways of specifying those methods which will be supported by every implementation; however, the syntax is capable of supporting the use of non-standard methods for private interchange.

28.2 Image Dictionaries

Both the **ImageRasterElement** and **MaskBitMap** operators take a reference to an *image Dictionary* as their single explicit operand. An image Dictionary is a self-contained collection of all the parameters necessary for imaging a sampled or bitmap raster graphics image. The contents of the image Dictionary specify the following:

— The source(s) of data for the raster graphics image

— The methods for extraction and interpretation of the data

— The mapping of the current User Coordinate Space to the image coordinate space

28.2.1 Image Dictionary Contents and Semantics

The required and optional image Dictionary key/value pairs are:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642732 101

28.2.1.1 DataSources

The mandatory key/value pair <*DataSources: Vector*> references a Vector of data sources. A detailed description of the required contents of *DataSources* in both single and multiple data source cases is given in 28.2.2 .

28.2.1.2 Width

The mandatory key/value pair <*Width: Cardinal*> specifies the width of the image or mask bitmap in samples.

28.2.1.3 Height

The mandatory key/value pair <*Height: Cardinal*> specifies the height of the image or mask bitmap in samples.

28.2.1.4 BitsPerComponent

The mandatory key/value pair <*BitsPerComponent. Cardinal*> specifies the number of bits used to represent each color component. The *BitsPerComponent* key/value pair specifies how to decompose the raster graphics image data into Cardinal sample values between 0 and (2 ° *BitsPerComponent*)-1. In decomposing the data, the **ImageRasterElement** and **MaskBitMap** operators read the octets of the data in the order in which they are obtained from the data source, and reads bits starting with the high-order bit of each octet. For each sample value, the first bit read is the high order bit of that sample value. Subsequent bits are the next lower order bits. Allowed values for *BitsPerComponent* are 1, 2, 4, 8, and 12 If the image Dictionary is an operand to the **MaskBitMap** operator, the value bound to the *BitsPerComponent* key shall be 1. Each row of sample values shall begin on an octet boundary If a row of sample values is not an integral number of octets long the data shall be padded with enough bits to fill the last octet.

28.2.1.5 Decode

The mandatory key/value pair <*Decode: VectorReference*> references a Vector called the Decode Vector. If the image Dictionary is an operand to the **ImageRasterElement** operator, the *Decode* Vector specifies the mapping of the sample values to the range of values appropriate to the corresponding color space components. If the image Dictionary is an operand to the **MaskBitMap** operator, the *Decode* Vector specifies the polarity of the mask. A detailed description of the required contents of the *Decode* Vector in each case is given in 28.2.4 .

28.2.1.6 Interpolate

The optional key/value pair <*Interpolate: Boolean*> specifies whether image interpolation is to be perfomed (if the implementation is capable of doing so). See 28.2 5. A value of true specifies that image interpolation is to be performed A value of false specifies no image interpolation is to be performed. If this key is not in the image Dictionary, the default value is false

28.2.1.7 ImageMatrix

The mandatory key/value pair <*ImageMatrix: Transformation*> specifies a transformation of the image. The raster graphics image is considered to exist in its own rectangular *image coordinate space* whose lower-left corner is (0,0) and whose upper-right corner is (*Width,Height*). The *rows* of a raster graphics image are the set of samples with the same first coordinate value. The *columns* of a raster graphics image are the set of samples with the same second coordinate value. *ImageMatrix* specifies the mapping from User Coordinate Space to this image coordinate space.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642733 048

28.2.2 Data Sources

The image Dictionary contains key/value pairs specifying the *Width* and *Height* of the raster graphics image data in samples. The sample or bitmap mask values acquired by the raster graphics operators during decomposition are interpreted as a rectangular array and are assumed to be acquired in a fixed order: (0,0) through (Width-1,0), then (0,1) through (Width-1,1), until (0, *Height*-1) through (*Width*-1, *Height*-1). It is the responsibility of the data sources specified in the *DataSources* image Dictionary key/value pair to provide octets of data in such a manner that decomposition yields correct sample values in this order. If the image Dictionary is used as the operand of a **MaskBitMap** operator, the value of the *DataSources* key in the image Dictionary shall be a reference to a single-element *DataSources* Vector.

28.2.2.1 Single-element DataSources Vector

If the value *DataSources* key in the image Dictionary is a reference to a single-element Vector, it is assumed to specify a single data source. If the current color space has more than one component, the data for all color components is assumed to be acquired from the data source in bit-wise order, interleaved on a per-sample basis. The set of allowable data sources types is:

— *StreamObject*

— *OctetString*

— *Procedure*

If the data source is of type *StreamObject*, there shall be sufficient data available to the decomposition process to produce the required number of samples. StreamObjects may be created in content by the **Filter** operator (see clause 31) or the **FindResource** operator (see clause 21). StreamObjects created by the **FindResource** operator may be filtered, depending on the Data Source resource bound to the INTERNAL RESOURCE IDENTIFIER operand to the **FindResource** operator, as follows:

— If the Data Source is in-line in the document (see 28.2.3), and the document is encoded using the Clear Text Representation and Interchange Format (see clause 37), the *StreamObject* created is filtered using the *ASCII85Decode* filter (see clause 31).

— If the Data Source is external, in either the Binary or Clear Text Representation and Interchange Format, or in-line in a document in the Binary Representaion and Interchange Format, the StreamObject created is not filtered.

If the data source is of type OctetString, the octets that make up the OctetString are taken in sequential order as octets of image or bitmap mask data. If the length of the OctetString is insufficient to provide the required number of samples, the data in the OctetString is re-used repeatedly until the required number of samples have been extracted.

If the data source is of type *Procedure*, when executed it shall return (on the Operand Stack) an OctetString containing octets of sampled raster graphics or bitmap mask data. If the data in the OctetString is exhausted before the required number of samples have been extracted, the Procedure is called repeatedly to return new OctetStrings of data until the required number of samples have been extracted. The Procedure may not execute the **ImageRasterElement** or **MaskBitMap** operators.

NOTE 161 The Procedure should not alter the contents of the image Dictionary or the graphics state; the effect of such actions is not defined in this International Standard and may cause unpredictable and undesirable results.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642734 T84

28.2.2.2 Multi-element DataSources Vector

If the value bound to the *DataSources* key in the image Dictionary is a reference to a multi-element Vector, it is assumed to specify multiple data sources; one data source per color component in the current color space. Data for each color component of the current color space is drawn from a different data source. All elements of the *DataSources* Vector shall be of the same type, and the ordering of the data sources in the Vector shall match the ordering of the required color components for the current color space.

In the multiple data sources case, each data source shall meet the requirements set forth above for the single data source case. In addition, the following restrictions also apply:

— If the data sources are of type *StreamObject*, they shall all be different *StreamObjects*.

— If the data sources are of type *OctetString*, they shall all be of the same length.

— If the data sources are of type *Procedure*, they are each called in turn and each shall return an *OctetString* of the same length on any given iteration.

28.2.3 In-Line Data

SPDL provides for the inclusion of raster graphics image data in documents, termed *in-line data*, via the *DataBlock* content token type This token type, represents a sequence of octets, and is defined for documents in both the Binary or Clear Text Representation and Interchange Format in clause 37 and clause 38. In documents in the Binary Representation and Interchange Format, these octet sequences represent binary raster graphics image data; in documents in the Clear Text Representation and Interchange Format the octet sequences represent *ASCII85*-encoded raster graphics image data (see clause 31 and 37.2.4).

The use of in-line raster graphics image data shall be as follows:

— In structure, use the Object Name *data-source::Document* to construct a suitable Public Object Identifier for the in-line Data Source pre-defined (see 9.7) by this International Standard) and bind this to the value of an INTERNAL RESOURCE IDENTIFIER via a RESOURCE DECLARATION structure element.

— In content within the scope of the RESOURCE DECLARATION, obtain a StreamObject by executing the **FindResource** operator using the value of the above INTERNAL RESOURCE IDENTIFIER as an operand.

— Construct an image Dictionary, using the StreamObject as the (or one of the) elements of the *DataSources Vector*, as appropriate. Only one in-line DataSource is allowed in a multi-element Vector.

— Construct a TOKENSEQUENCE or sequence of TOKENSEQUENCES containing an **ImageRasterElement** or **MaskBitMap** raster graphics imaging operator that uses the in-line data as a data source.

— Immediately follow the operator which causes the **ImageRasterElement** or **MaskBitMap** operator to be interpreted with all data needed by the imaging operator, in the form of one *DataBlock* token.

In-line data shall be placed in content so as to be accessible by the raster graphics imaging operator when the operator is actually interpreted. Thus, if the raster graphics imaging operator occurs inside a Procedure which is interpreted by a looping operator (for example, the **For** operator), the in-line data for all interpretations of the raster graphics imaging operator in the loop shall immediately follow the looping operator in the form of one or more *DataBlock* tokens. The order of these sequences shall correspond to the sequential order of interpretation of the raster graphics imaging operator in the loop, and the data for each interpretation of the raster graphics imaging operator shall begin with a new *DataBlock* token.

A *DataBlock* token encountered in content other than where necessary as a data source for an imaging operator shall result in **RaiseError** being invoked with *SyntaxError* as its operand.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

28.2.4 Decode Vector

In an image Dictionary used as an operand to the **ImageRasterElement** operator, the value bound to the *BitsPerComponent* key in the image Dictionary specifies how to decompose the raster graphics image data into sample values. The *Decode* Vector specifies a linear mapping of the sample values into the range of values appropriate to each color component of the current color space. For each color component, a minimum and maximum value for the mapping are specified, and the linear mapping will be

$$OutputValue = MIN + InputValue \times (MAX - MIN) / N$$

where

— *OutputValue* = the mapped value to be used as the final color component,

— *InputValue* = the sample value taken from the data source during decomposition,

— N = (2 × *BitsPerComponent*)-1, and

— MIN and MAX are the parameters specified in the *Decode* Vector.

The numbers in the *Decode* Vector are interpreted in pairs, with each pair applying to the one component of the current color space. Thus the length of the *Decode* Vector shall be twice the number of color components in the current color space, and the ordering of the MIN/MAX pairs in the *Decode* Vector shall match the ordering of the required color components for the current color space. If an OutPutValue falls beyond the valid range for the color component, the OutputValue will be set to the nearest value in the valid range.

In an image Dictionary used as an operand to the **MaskBitMap** operator, the Decode Vector is a two-element Vector which specifies the polarity of the mask. The two allowed values of the Decode Vector in this case, and their meanings, are shown in table 33:

Table 33

Value	meaning
[0 1]	Sample values of 0 are painted with the current color; sample values of 1 correspond to the opaque part of the mask
[1 0]	Sample values of 1 are painted with the current color; sample values of 0 correspond to the opaque part of the mask

28.2.5 Interpolation

When the resolution of the source image is significantly lower than the resolution of the presentation device, each source sample covers many pixels of the device; this can result in poor output appearance. These visual artifacts can be reduced by interpolation between adjacent sample values. If the value bound to the *Interpolate* key in an image Dictionary is true, this is an indication to the implementation that interpolation is desired. There are, however, no interpolation algorithms defined by this International Standard.

NOTE 162 The nature and implementation of interpolation algorithms can depend on the device on which the output of processing is being realized. Hence, the choice of interpolation algorithm, if any, is left to the implementation.

28.3 Operators

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642736 857

exception handling are defined in clause 36. In addition to these operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6 2.

28.3.1 ImageRasterElement

The **ImageRasterElement** imaging operator accepts one explicit operand

<imagedictionary: DictionaryReference>

and returns no results. The **ImageRasterElement** operator uses the information in the *imagedictionary* to acquire sample values, interpret them, and image the raster graphics image data.

The set of mandatory key/value pairs is checked. If any keys are missing, **RaiseError** shall be invoked with *UndefinedKey* as its operand. If the key is present but the value bound to it is of the wrong type, **RaiseError** shall be invoked with *TypeCheck* as its operand. If the key is present and the type is correct but the bound value is out of range, **RaiseError** shall be invoked with *RangeCheck* as its operand.

If any *DataSources* are Procedures which execute the **ImageRasterElement** or **MaskBitMap** operator, **RaiseError** shall be invoked with *UndefinedKey* as its operand. If there are multiple *DataSources* and any one terminates before the others (either due to an End-Of-Data condition on a StreamObject or due to a Procedure returning a empty OctetString), **RaiseError** shall be invoked with *RangeCheck* as its operand. If there is only one element in *DataSources* and it terminates before the complete rectangular array is imaged, then **RaiseError** will not be invoked and the page image will have that portion of the array that was imaged except that the last row imaged may not be present if it was not a full row. If there are problems in any StreamObject that is an element in *DataSources*, **RaiseError** shall be invoked with *IOError* or *DataError*, as appropriate, as its operand.

The input to the imaging activity is a sequence of sample values to be imaged as a rectangular array. Each sample value corresponds to a unit rectangle in the image coordinate system described in the *ImageMatrix* key/value pair in the *imagedictionary*. The order of the sample values is specified in 28.2.2.

The inverse of the *ImageMatrix* transformation transforms and positions the image coordinate space into the UCS.

NOTE 163 It is often simpler to position a sampled raster graphic image on a page by splitting the positioning into three simple transformations. First the *ImageMatrix* transformation is defined to map the unit square in UCS into the image coordinate space and to handle the scanline order of the image data. A *Width* by *Height* image that was scanned from left to right within a line and from top to bottom among the lines (the first sample is in the top left corner) would have an *ImageMatrix* of [*Width* 0 0 -*Height* 0 *Height*]. Then, before the sampled raster graphic image is imaged, UCS is changed by {LowerLeftX LowerLeftY **Translate**XSizeinUCS YSizeinUCS **Scale**}. The first operator positions the lower left corner of the sampled raster graphic image and the second determines the size of that image in the prior UCS. To preserve the prior UCS, encompass these transformations and the **ImageRasterElement** operator between a **SaveGraphicsState** operator and a **RestoreGraphicsState** operator.

The *CurrentColorSpace* determines the number of color components per sample value. The vector of potential color components that make up each sample value is assembled by using the *BitsPerComponent* key/value pair in *imagedictionary* to decompose the OctetStrings produced as described in 28.2.2. The *Decode* key/value pair in *imagedicationary* is used to map the potential color components into actual color components which can be used to specify a color in the *CurrentColorSpace* as described in 28.2.4.

The array of colored rectangular areas is imaged onto the page image, element by element. Each element is painted onto the page image using the color of that element as the ink, the rectangular area the element occupies as the mask, and the *CurrentClippingRegion* as the clipping region.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642737 793

28.3.2 MaskBitMap

The *MaskBitMap* imaging operator accepts one explicit operand

<imagedictionary: DictionaryReference>

and returns no results. The **MaskBitMap** operator uses the information in the *imagedictionary* to acquire bitmap sample values, interpret them to form a mask, and execute an imaging activity using the *CurrentColor* Graphics State Variable and the mask The contents and semantics of image Dictionaries are described in detail in 28.2.

The set of mandatory key/value pairs is checked. If any keys are missing, **RaiseError** shall be invoked with *UndefinedKey* as its operand. If the key is present but the value bound to it is of the wrong type, **RaiseError** shall be invoked with *TypeCheck* as its operand. If the key is present and the type is correct but the bound value is out of range, **RaiseError** shall be invoked with *RangeCheck* as its operand. If the sole element of *DataSources* is a Procedure which executes the **ImageRasterElement** or **MaskBitMap** operator, **RaiseError** shall be invoked with *UndefinedKey* as its operand.

If the single element in *DataSources* is a Procedure which executes the *ImageRasterElement* or *MaskBitMap* operator, **RaiseError** shall be invoked with *UndefinedKey* as its operand. If there are problems in any StreamObject that is an element in *DataSources*, **RaiseError** shall be invoked with *IOError* or *DataError*, as appropriate, as its operand.

The input to the mask creation is a sequence of sample values which are positioned in a rectangular array. Each sample value corresponds to a unit rectangle in the image coordinate system described in the *ImageMatrix* key/value pair in the *imagedictionary*. The order of the sample values is specified in 28.2.2.

The inverse of the *ImageMatrix* transformation transforms and positions the image coordinate space into the UCS.

The *Decode* key/value pair in *imagedictionary* is used to determine whether the unit rectangle that corresponds to the sample value is part of the mask or part of the opaque region.

The mask thus created is used in an imaging activity in which the ink is the *CurrentColor* in the *CurrentColorSpace* and the *CurrentClippingRegion* is the clipping region.

29 Geometric Graphics Operators

29.1 Model for Geometric Graphics

Shapes are defined geometrically in terms of paths. Paths are composed of path segments, which in turn are composed of path elements as defined in 18.2.3 .

Two path elements are *connected* (1) if they are in sequence in any path segment or (2) if they are the initial and final path elements of a closed path segment. The point at which two path elements connect is called a *connection point*.

A *closed path segment* is one for which the end point of the last path element coincides with and is connected to the start point of the initial path element. A *closed path* is one consisting solely of closed path segments.

Paths are constructed by primitive operators and retained in the Graphics State Variable *CurrentPath*. A path segment is begun by **SetPosition** and may be extended by operators such as **LineTo** and **CurveTo**. The *CurrentPath* can be obtained by **GetPath** and set by **SetPath**, and is cleared to the null path by the **NewPath** operator or by the imaging and clipping operators. Several of the imaging operators, **RectangleFill** and

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642738 62T

RectangleStroke, and the clipping operator **RectangleClip** construct a path in conjunction with their other activities.

Paths are given geometrical interpretation only when they are used to specify a mask or a clipping region. At this time the points which define the path are interpreted as defining geometry in the User Coordinate System. The imaging operators **StrokePath**, **FillPath**, **FillPathEvenOdd**, **RectangleFill**, **RectangleStroke**, and the clipping operators **ClipPath**, **ClipPathEven** and **RectangleClip** convert this geometry to the Reference Coordinate System by applying the *CurrentTransformation*. The geometric shapes which result from the execution of these operators are specified by the operator definitions.

29.2 Path Construction and Manipulation Operators

This subclause describes the operators used to construct and directly manipulate paths. All path construction operators that affect the value of the *CurrentPath* Graphics State Variable also implicitly set the value of the *CurrentPosition* Graphics State Variable to the end point of the current path. A null path has no current point and when a null path is created, the value of the *CurrentPosition* Graphics State Variable is undefined.

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and exception handling are defined in clause 36. In addition to these operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6.2.

29.2.1 AppendPath

The **AppendPath** imaging operator takes one operand

<path: PathReference>

and returns no results. It appends a copy of the Path referenced by *path* to the Path referenced by the value of the *CurrentPath* Graphics State Variable. The *CurrentPosition* Graphics State Variable is set to the end point of the last path segment in the appended Path.

29.2.2 ArcToClockwise

The **ArcToClockwise** imaging operator takes five operands

<t_2: Number>

<t_1: Number>

<r: Number>

<y: Number>

<x: Number>

and returns no results. It creates a new path element consisting of a directed circular arc whose

- center is (x, y);
- radius is r;
- start point is the point whose polar coordinates relative to (x, y) are (r, t_1);

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642739 566

— end point is the point whose polar coordinates relative to (x, y) are (r, t_2),

— direction is clockwise.

The *CurrentPosition* Graphics State Variable is set to that end point.

If the *CurrentPath* is the null path, **ArcToClockwise** begins a new path segment consisting of the start point of the new path element followed by the new path element.

If the *CurrentPath* is not the null path, and if the start point of the new path element coincides with the end point of the last segment of the *CurrentPath*, the path element is appended to that path segment. Otherwise, if the start point of the new path element does not coincide with the end point of the last segment of the *CurrentPath*, **ArcToClockwise** appends to the *CurrentPath* a line segment whose start point is the end point of the last segment of the *CurrentPath* and whose end point is the start point of the new path element, then appends the new path element to the line segment.

29.2.3 ArcToCounterClockwise

The **ArcToCounterClockwise** imaging operator has the same semantics as the **ArcToClockwise** operator (see 29.2.8), but draws a counterclockwise arc. Specifically, the **ArcToCounterClockwise** operator takes five operands

<t_2: *Number*>

<t_1: *Number*>

<*r: Number*>

<*y: Number*>

<*x: Number*>

and returns no results. It creates a new path element consisting of a directed circular arc whose

— center is (x, y);

— radius is r;

— start point is the point whose polar coordinates relative to (x, y) are (r, t_1);

— end point is the point whose polar coordinates relative to (x, y) are (r, t_2),

— direction is counterclockwise.

The *CurrentPosition* Graphics State Variable is set to that end point.

If the *CurrentPath* is the null path, **ArcToCounterClockwise** begins a new path segment consisting of the start point of the new path element followed by the new path element.

If the *CurrentPath* is not the null path, and if the start point of the new path element coincides with the end point of the last segment of the *CurrentPath*, the path element is appended to that path segment. Otherwise, if the start point of the new path element does not coincide with the end point of the last segment of the *CurrentPath*, **ArcToCounterClockwise** appends to the *CurrentPath* a line segment whose start point is the end point of the last segment of the *CurrentPath* and whose end point is the start point of the new path element, then appends the new path element to the line segment.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

29.2.4 ClosePathSegment

The **ClosePathSegment** imaging operator takes no operand and returns no results. If the current path segment is already closed, or if *CurrentPath* is the null path, **ClosePathSegment** has no effect. Otherwise, if ($cpsb_x$, $cpsb_y$) are the UCS coordinates of the start point of the last path segment of the *CurrentPath*, **ClosePathSegment** is equivalent to {$cpsb_x$ $cpsb_y$ **LineTo** }, with the additional property that the initial path element of the last path segment is connected to the new final path element of that path segment at the start point of the initial path element and the end point of the new path element. The *CurrentPosition* Graphics State Variable is set to the point with UCS coordinates ($cpsb_x$, $cpsb_y$).

NOTE 164 A closed path does not result from causing the end point of the last path element of a path segment to coincide with the start point of that path segment; such a path is still an open path and, when used with the **StrokePath** operator, appropriate stroke ends will be realized at the start and end points of that path segment (see 29.3.3). For round stroke ends, the result may be indistinguishable from round joins applied to the connection points of the path segment.

Appending another path element to the *CurrentPath* immediately after a **ClosePathSegment** operator will begin a new path segment, even if it is drawn from the end point reached by **ClosePathSegment**.

29.2.5 CurveTo

The **CurveTo** imaging operator takes six operands

<y_3: *Number*>

<x_3: *Number*>

<y_2: *Number*>

<x_2: *Number*>

<y_1: *Number*>

<x_1: *Number*>

and returns no results. It appends a new path element consisting of a Bézier curve to the last path segment of the *CurrentPath* Graphics State Variable. The start point of this path element is the end point of the last path segment of the *CurrentPath*. The end point of the path element has UCS coordinates (x_3, y_3). The *CurrentPosition* Graphics State Variable is set to that end point. The path element has two Bézier control points whose UCS coordinates are (x_1, y_1) and (x_2, y_2).

More precisely, if the end point on the last path segment of *CurrentPath* is (x_0, y_0), the new path element consists of the points on the parametric curve represented by the equations

$$x = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y = a_y t^3 + b_y t^2 + c_y t + d_y$$

for 0 σ t σ 1, where

$$a_x = x_3 - 3x_2 + 3x_1 - x_0$$

$$a_y = y_3 - 3y_2 + 3y_1 - y_0$$

$$b_x = 3x_2 - 6x_1 + 3x_0$$

$$b_y = 3y_2 - 6y_1 + 3y_0$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642741 114

$c_x = 3x_1 - 3x_0$

$c_y = 3y_1 - 3y_0$

$d_x = x_0\ d_y = y_0$

NOTE 165 the new path element is tangent to the line segment from (x_0, y_0) to (x_1, y_1) at its start point and is tangent to the line segment from (x_2, y_2) to (x_3, y_3) at its end point, provided those points are distinct.

If *CurrentPath* is the null path, **RaiseError** shall be invoked with *NoCurrentPosition* as its operand.

29.2.6 CurveToRelative

The **CurveToRelative** imaging operator takes six operands

<y_3: *Number*>

<x_3: *Number*>

<y_2: *Number*>

<x_2: *Number*>

<y_1: *Number*>

<x_1: *Number*>

and returns no results. If (*cpsx, cpsy*) is the UCS point which corresponds to the end point of the last path segment of the *CurrentPath*, **CurveToRelative** appends a new path element consisting of a Bézier curve to the last path segment of the *CurrentPath* Graphics State Variable. The start point of this path element is the end point of the last path segment of the *CurrentPath*. The end point of the path element has UCS coordinates $(cpsx + x_3, cpsy + y_3)$. The *CurrentPosition* Graphics State Variable is set to that end point. The path element has two Bézier control points whose UCS coordinates are $(cpsx + x_2, cpsy + y_2)$ and $(cpsx + x_3, cpsy + y_3)$. The equations for this curve are as given in 29.2.5.

If *CurrentPath* is the null path, **RaiseError** shall be invoked with *NoCurrentPosition* as its operand.

29.2.7 GetPath

The **GetPath** imaging operator takes no operands and returns one result

<*path: PathReference*>

where *path* is a reference to a copy of the Path that is referenced by the value of the *CurrentPath* Graphics State Variable. *CurrentPath* is not affected by **GetPath**.

29.2.8 GlyphToPath

The **GlyphToPath** imaging operator takes two operands

<*fill: Boolean*>

<*glyphstr: GlyphString*>

and returns no results. **GlyphToPath** appends to the *CurrentPath* a set of path segments which describe the glyph shapes as they would be presented by the **ShowString** operator (see clause 27). If the value of *fill* is true,

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642742 050

the path segments are constructed in such a way as to make them suitable for filling or clipping via the **FillPath** or one of the **ClipPath** operators. If the value of *fill* is false, the path segments are constructed in such a way as to make them suitable only for being stroked via the **StrokePath** operator. The *CurrentPosition* Graphics State Variable is set to the end point of the last generated path segment added to the *CurrentPath*.

NOTE 166 The effect of applying the **StrokePath** operator to a path suitable for filling, or of applying the **FillPath** or one of the two **ClipPath** operators to a path suitable for stroking, is not defined by this International Standard and may result in an exception being raised.

If *CurrentPath* is the null path, **RaiseError** shall be invoked with *NoCurrentPosition* as its operand.

29.2.9 LineTo

The **LineTo** imaging operator takes two operands

<y: Number>

<x: Number>

and returns no results. It appends a new path element consisting of a straight line segment, whose start point is the end point of the last path segment of the *CurrentPath* and whose end point has UCS coordinates (*x*, *y*), to the *CurrentPath* Graphics State Variable. The *CurrentPosition* Graphics State Variable is set to that end point.

If *CurrentPath* is the null path, **RaiseError** shall be invoked with *NoCurrentPosition* as its operand.

29.2.10 LineToRelative

The **LineToRelative** imaging operator takes two operands

<y: Number>

<x: Number>

and returns no results If (*cpsx*, *cpsy*) is the UCS point which corresponds to the end point of the last segment of the *CurrentPath*, **LineToRelative** appends a new path element to the last segment of the *CurrentPath* consisting of a straight line segment, whose start point is the end point of the last path segment of the *CurrentPath* and whose end point has UCS coordinates (*cpsx* + *x*, *cpsy* + *y*). The *CurrentPosition* Graphics State Variable is set to that end point

If *CurrentPath* is the null path, **RaiseError** shall be invoked with *NoCurrentPosition* as its operand.

29.2.11 NewPath

The **NewPath** operator takes no operands and returns no results. It sets the Graphics State Variable *CurrentPath* to the null path, and makes the value of the *CurrentPosition* Graphics State Variable undefined.

29.2.12 OutlineStroke

The **OutlineStroke** imaging operator takes no operands and returns no results. It replaces the value of the *CurrentPath* Graphics State Variable with a path that exactly encloses the shape that would result if the **StrokePath** operator were applied to the *CurrentPath*. The path resulting from **OutlineStroke** is suitable as the implicit operand to **FillPath** or **ClipPath**. In general, this path is not suitable for **StrokePath**, as it may contain interior path elements or disconnected path segments that are produced by **OutlineStroke**'s stroke to outline conversion process. If **OutlineStroke**applied to the *CurrentPath* would not affect the page image, then the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642743 T97

CurrentPath is set to the null path, and the value of the *CurrentPosition* Graphics State Variable shall be undefined.

29.2.13 SetPath

The **SetPath** imaging operator takes one operand

<*path: Path*Reference>

and returns no results. It sets the value of the *CurrentPath* Graphics State Variable to a reference to a copy of the Path referenced by *path*. The *CurrentPosition* Graphics State Variable is set to the end point of the last path segment of the Path referenced by *path*.

29.3 Stroking and Filling Operators

This subclause describes the operators used to stroke and fill the regions described by paths The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and exception handling are defined in clause 36. In addition to these operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6.2.

29.3.1 FillPath

The **FillPath** imaging operator accepts no operands and returns no results. It performs an imaging activity using a mask defined using the *CurrentPath* Graphics State Variable. It sets the value of *CurrentPath* to the null path and makes the *CurrentPosition* Graphics State Variable be undefined.

In performing the imaging activity, **FillPath** implicitly closes any open path segments and defines a mask which includes the interior and boundary of the resulting shape determined according to the non-zero winding rule. The mask is then used with the *CurrentColor* and *CurrentClippingRegion* to add an image element to the current page image.

The effect of executing the **FillPath** operator when the *CurrentPath* consists of a single line segment is equivalent to {0 **SetStrokeWidth StrokePath**}.

29.3.2 FillPathEvenOdd

The **FillPathEvenOdd** imaging operator accepts no operands and returns no results. It performs an imaging activity using a mask defined using the *CurrentPath* Graphics State Variable. It sets the value of *CurrentPath* to the null path and makes the *CurrentPosition* Graphics State Variable be undefined.

In performing the imaging activity, **FillPathEvenOdd** implicitly closes any open path segments and defines a mask which includes the interior and boundary of the resulting shape determined according to the even/odd winding rule. The mask is then used with the *CurrentColor* and *CurrentClippingRegion* to add an image element to the current page image.

The effect of executing the **FillPathEvenOdd** operator when the *CurrentPath* consists of a single line segment is equivalent to {0 **SetStrokeWidth StrokePath**}.

29.3.3 StrokePath

The **StrokePath** imaging operator accepts no operands and returns no results. It performs an imaging activity using a mask defined by *CurrentPath*, *CurrentDashPattern*, *CurrentMiterLimit*, *CurrentStrokeAdjust*,

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642744 923

CurrentStrokeWidth, *CurrentStrokeJoin*, and *CurrentStrokeEnd* as specified below. It sets the value of *CurrentPath* to the null path and makes the *CurrentPosition* Graphics State Variable be undefined.

In performing the imaging activity, **StrokePath** constructs a mask in the User Coordinate Space, transforms the mask to Reference Coordinate Space using the *CurrentTransformation*, and paints an image element using this mask, the *CurrentColor*, and the *CurrentClippingRegion*. In performing these functions, **StrokePath**

— transforms the *CurrentPath* to User Coordinate Space using the inverse of the *CurrentTransformation*;

— constructs a shape in the User Coordinate System as described below;

— transforms this shape to Reference Coordinate Space using the *CurrentTransformation*;

— paints an image element using

- a mask whose interior and boundary is the shape described above,
- the *CurrentColor*,
- the *CurrentClippingRegion*.

The shape constructed in the second step is either a *solid line* or a *dashed line* along each of the path segments, depending on the value of *CurrentDashPattern* (see figure 11). If the Vector component of *CurrentDashPattern* is the empty Vector, the shape is constructed by

— widening each path segment to width *CurrentStrokeWidth* uniformly in the UCS in such a way that the resultant widened line is longitudinally bisected by the original path segment;

— applying the stroke end features specified by *CurrentStrokeEnd* (as shown in figure 12) to the ends of each path segment;

— applying the join features specified by *CurrentStrokeJoin* (as shown in figure 13), and possibly as modified by *CurrentMiterLimit* (see 29.3.3.1) to connection points of a path segment.

If the value of the *CurrentStrokeWidth* Graphics State Variable is 0, the actual stroke width used is the smallest width the presentation device is capable of producing.

The actual width of the stroke may also differ from the value of *CurrentStrokeWidth* if the value of the *CurrentStrokeAdjust* Graphics State Variable is true; If the value of *CurrentStrokeAdjust* is true, the stroke width and the coordinates of a stroke are automatically adjusted as necessary to produce lines of uniform thickness, and whose widths are no more than $\pm$ half a pixel different from the requested width (as determined by the value of the *CurrentStrokeWidth* Graphics State Variable).

If the Vector component of *CurrentDashPattern* is non-empty, a new path is constructed in User Coordinate System and a shape is constructed using that path as described above.

If the value of the *CurrentDashPattern* Graphics State Variable is

<v: Vector><x: Number>

then the new path is obtained from the transformation of the *CurrentPath* to User Coordinate System generated in the first step as follows:

— The Vector *v* is interpreted as a sequence of distances along the transformation of the *CurrentPath* to User Coordinate Space. These distances alternately specify the length of a dash and the length of a gap between

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642745 86T

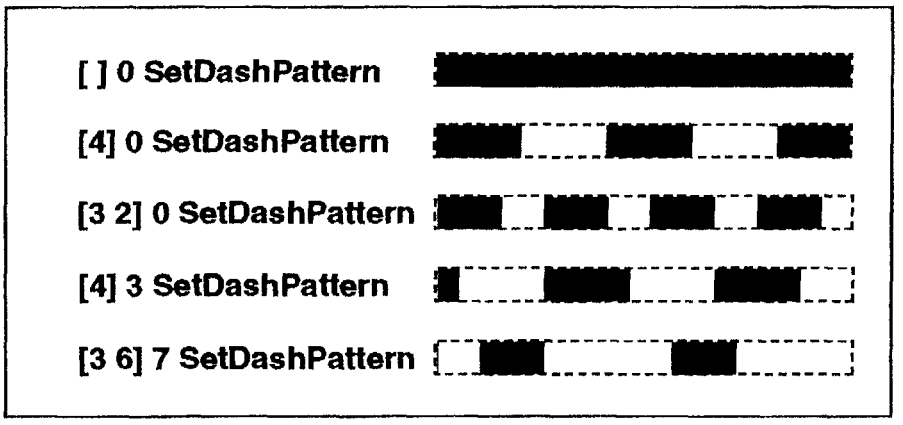

Figure 11 — Examples of Dash Patterns

dashes The contents of *v* are used cyclically for as many values and as many cycles as needed (see Figure 11).

— The Number *x* is interpreted as an offset into the sequence of distances specified by the cyclic use of *v*. Before beginning to stroke a path, the elements of *v* are cycled through adding up the sum of the elements (dash lengths and gap lengths) until a total distance of *x* has been used. The lengths used to generate the dash pattern are the remainder of the element after the offset x has been subtracted from it plus the succeeding lengths taken from the cyclic processing of *v*.

— The new path consists of a sequence of path segments which are those parts of the transformation of the *CurrentPath* to User Coordinate Space which correspond to the dash elements obtained from *CurrentDashPattern*.

If the result of the transformation of the *CurrentPath* to User Coordinate Space contains no displacement from the start point of the path, the path is said to have zero length. Stroking a path with no path elements does not affect the page image. Thus the code fragment {**NewPath** *x y* **SetPosition StrokePath**} produces no marks on the current page. Stroking a zero length path will also not affect the page image unless the *CurrentStrokeEnd* Graphics State Variable specifies round stroke ends; in this case the stroke ends will appear when the path is stroked.

29.3.3.1 Effect of CurrentMiterLimit on Stroke Joins

At any given corner (a connection point between two non-tangential path elements), the *miter length* is the distance from the point at which the inner edges of the stroke (as constructed in User Coordinate System by the **StrokePath** operator) intersect to the point at which the outside edges of the stroke intersect. This distance increases as the angle between the path elements decreases.

The *CurrentMiterLimit* Graphics State Variable value is employed to prevent this distance, the diagonal length of the miter, from exceeding a desired maximum length when the value of the *CurrentStrokeJoin* specifies that the **StrokePath** operator use miter joins. When the ratio of the miter length of a given join to the actual stroke width exceeds the *CurrentMiterLimit*, the **StrokePath** operator constructs the stroke for the join as if the value of the *CurrentStrokeJoin* Graphics State Variable specified a bevel join rather than a miter join. The formula for the ratio of the miter length to the stroke width is

$$\textit{miter length} / \textit{stroke width} = 1 / \sin(\textit{angle}/2)$$

where *angle* is the angle of the intersection of the two path elements.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642746 7T6

29.4 Rectangular Fill and Stroke Operators

There are two operators which combine defining a path and either stroking or filling that path.

29.4.1 RectangleFill

The **RectangleFill** imaging operator takes four operands

<height. Number>

<width: Number>

<y: Number>

<x: Number>

and returns no results. It creates a new closed rectangular path with one path segment, fills its and destroys the path all without reading or altering the *CurrentPath* Graphics State Variable or the *CurrentPosition* Graphics State Variable

Under the assumption that *height* and *width* are positive, the **RectangleFill** operator is equivalent to the following procedure:

```
{
    SaveGraphicsState
% Start a new path for the rectangle
    NewPath
% Position at one corner
    x y SetPosition
% Draw lines to other corner
    width 0 LineToRelative
    0 height LineToRelative
% Add final two sides of rectangle
    width Negate 0 LineToRelative
    ClosePathSegment
% Fill the result
    FillPath
% Erase all trace of the rectangle path
    RestoreGraphicsState
}
```

29.4.2 RectangleStroke

The **RectangleStroke** imaging operator takes either four operands

<height: Number>

<width: Number>

<y. Number>

<x: Number>

or five operands

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642747 632

<matrix: Transformation>

<height: Number>

<width: Number>

<y: Number>

<x: Number>

and returns no results. It creates a new closed rectangular path with one path segment, strokes its and destroys the path all without reading or altering the *CurrentPath* Graphics State Variable or the *CurrentPosition* Graphics State Variable.

If the *matrix* operand is present, the **RectangleStroke** operator concatenates *matrix* to the CTM after defining the rectangular path, but before stroking that path. The *matrix* applies to the strokewidth and dash pattern, if any, but not to the path itself.

Under the assumption that *height* and *width* are positive, the **RectangleStroke** operator is equivalent to the following procedure:

```
{
   SaveGraphicsState
% Start a new path for the rectangle
   NewPath
%Position at one corner
   x y SetPosition
% Draw lines to other corner
   width 0 LineToRelative
   0 height LineToRelative
% Add final two sides of rectangle
   width Negate 0 LineToRelative
   ClosePathSegment
% Change CTM if matrix operand present
   matrix Concat
% Stroke the result
   StrokePath
% Erase all trace of the rectangle path
   RestoreGraphicsState
}
```

29.5 Graphics State Variable Operators

This subclause describes the operators used to obtain and affect the values of the Graphics State Variables that control various parameters of the stroking and filling processes.

29.5.1 GetDashPattern

The **GetDashPattern** operator takes no operands and returns two results

<offset: Number>

<pattern: VectorReference>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642748 579

where *pattern* is a reference to a Vector in which the elements are of type *Number* and the two results together form the current value of the *CurrentDashPattern* Graphics State Variable in the Graphics State.

29.5.2 GetMiterLimit

The **GetMiterLimit** operator takes no operands and returns one result

<limit: Number>

where *limit* is the value of the *CurrentMiterLimit* Graphics State Variable in the Graphics State.

29.5.3 GetStrokeAdjust

The **GetStrokeAdjust** operator takes no operands and returns one result

<adjust: Boolean>

where *adjust* is the value of the *CurrentStrokeAdjust* Graphics State Variable in the Graphics State.

29.5.4 GetStrokeEnd

The **GetStrokeEnd** operator takes no operands and returns one result

<end: Number>

where *end* is the value of the *CurrentStrokeEnd* Graphics State Variable in the Graphics State.

29.5.5 GetStrokeJoin

The **GetStrokeJoin** operator takes no operands and returns one result

<join: Number>

where *join* is the value of the *CurrentStrokeJoin* Graphics State Variable in the Graphics State.

29.5.6 GetStrokeWidth

The **GetStrokeWidth** operator takes no operands and returns one result

<width: Number>

where *width* is the value of the *CurrentStrokeWidth* Graphics State Variable in the Graphics State.

29.5.7 SetDashPattern

The **SetDashPattern** operator takes two operands

<offset: Number>

<pattern: VectorReference>

where *pattern* is a reference to a Vector in which the elements are of type *Number* and returns no results. It sets the*CurrentDashPattern* Graphics State Variable to *<pattern><offset>*. The effect of *CurrentDashPattern* is defined by the semantics of the **StrokePath** operator.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642749 405

29.5.8 SetMiterLimit

The **SetMiterLimit** operator takes one operand

<x: Number>

and returns no results. It sets the *CurrentMiterLimit* Graphics State Variable to *x*, where *x* shall have value greater than or equal to 1. The effect of *CurrentMiterLimit* is defined in 29 3 3.1.

29.5.9 SetStrokeAdjust

The **SetStrokeAdjust** operator takes one operand

<adjust: Boolean>

and returns no results. It sets the value of the *CurrentStrokeAdjust* Graphics State Variable to the value of *adjust*. The effect of *CurrentStrokeAdjust* is defined in the semantics of the **StrokePath** operator.

29.5.10 SetStrokeEnd

The **SetStrokeEnd** operator takes one operand

<n: Cardinal>

where $n = 0$, 1, or 2, and returns no results. It sets the *CurrentStrokeEnd* Graphics State Variable to one of the following as shown in table 34.

Table 34

Value	stroke ends
0	Selects butt stroke ends. The stroke is squared off at the end point of the path; there is no projection beyond the path
1	Selects round stroke ends. A circle with diameter equal to the line width is drawn around the end point and filled in
2	Selects projecting square stroke ends. The stroke continues beyond the end point of the path for a distance equal to half the line width, and is squared off

The effect of *CurrentStrokeEnd* on the mask generated by **StrokePath** is shown in figure 12.

29.5.11 SetStrokeJoin

The **SetStrokeJoin** operator takes one operand

<n: Cardinal>

where n = 0, 1, or 2, and returns no results. It sets the *CurrentStrokeJoin* Graphics State Variable to one of the following as shown in table 35.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642750 127

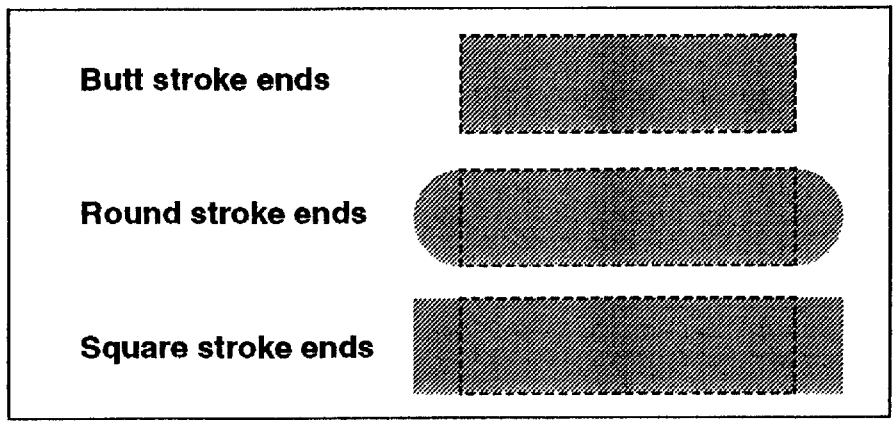

Figure 12 — Stroke End Features

Table 35

Value	miter join
0	Selects miter join. The outer edges of the strokes are extended until they meet
1	Selects round join. A circle with diameter equal to the *CurrentStrokeWidth* is drawn around the point where the path elements connect and is filled in
2	Selects bevel join. The connected path elements are finished with butt stroke ends; then the triangle defined by the outside endpoints of the butt stroke ends and the connection point is filled

The effect of *CurrentStrokeJoin* on the mask generated by **StrokePath** is shown in figure 13.

29.5.12 SetStrokeWidth

The **SetStrokeWidth** operator takes one operand

<x: Number>

and returns no results. It sets the *CurrentStrokeWidth* Graphics State Variable to *x*. The effect of *CurrentStrokeWidth* is defined by the **StrokePath** operator

30 Clipping Operators

30.1 Model for Clipping

The region of the page image within which imaging operators are allowed to have an effect is called the *clipping region*. The *CurrentClippingRegion* Graphics State Variable represents the clipping region in effect, and is represented in the Reference Coordinate System. At the beginning of a page, the *CurrentClippingRegion* is set to include the entire imageable region of the medium provided a superior block does not changed the *CurrentClippingRegion*. The operators which modify the *CurrentClippingRegion* do so by intersecting it with a region derived from the *CurrentPath*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642751 063

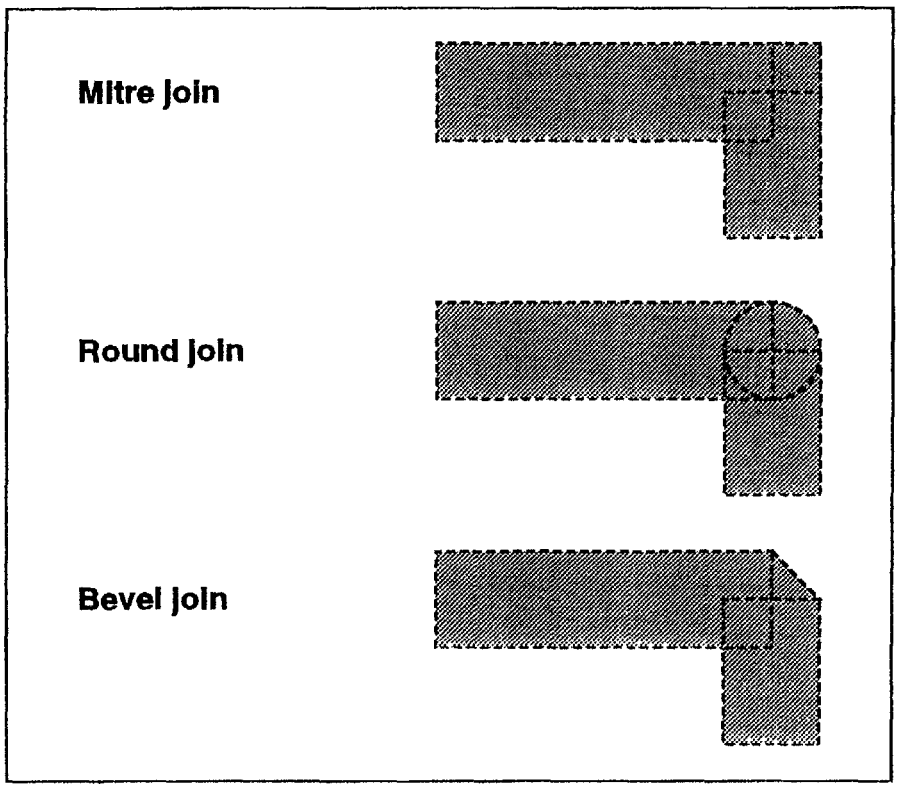

Figure 13 — Stroke Join Features

30.2 Operators

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and exception handling are defined in clause 36. In addition to these operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6.2.

30.2.1 ClipPath

The **ClipPath** operator accepts no operand and returns no results. It modifies the *CurrentClippingRegion* Graphics State Variable using the *CurrentPath* Graphics State Variable, but does not change the value of *CurrentPath*.

ClipPath implicitly closes any open path segments of the *CurrentPath* and determines the interior of the resulting shapes according to the non-zero winding rule. The Graphics State Variable *CurrentClippingRegion* is then set to the intersection of the current *CurrentClippingRegion* with the interior of the shape determined from the *CurrentPath*. If the *CurrentPath* is the null path, the value of the resultant *CurrentClippingRegion* is an implementation-dependent non-empty Path which encloses zero area.

30.2.2 ClipPathEvenOdd

The **ClipPathEvenOdd** imaging operator accepts no operand and returns no results. It modifies the *CurrentClippingRegion* Graphics State Variable using the *CurrentPath* Graphics State Variable, but does not change the value of *CurrentPath*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642752 TTT

ClipPathEvenOdd implicitly closes any open path segments of the *CurrentPath* and determines the interior of the resulting shapes according to the even/odd winding rule. The Graphics State Variable *CurrentClippingRegion* is then set to the intersection of the current *CurrentClippingRegion* with the interior of the shape determined from the *CurrentPath*. If the *CurrentPath* is the null path, the value of the resultant *CurrentClippingRegion* is an implementation-dependent non-empty Path which encloses zero area.

30.2.3 RectangleClip

The **RectangleClip** operator takes four operands

<height: Number>

<width. Number>

<y: Number>

<x: Number>

and returns no results. It creates a new closed rectangular path with one path segment. The Graphics State Variable *CurrentClippingRegion* is then set to the intersection of the value of the *CurrentClippingRegion* Graphics State Variable with the interior of the rectangular path. After compute the new clipping path, **RectangleClip** resets the *CurrentPath* Graphics State Variable to the null path and makes the *CurrentPosition* Graphics State Variable be undefined.

Under the assumption that *height* and *width* are positive, the **RectangleClip** operator is equivalent to the following procedure:

```
{
% Start a new path for the rectangle
   NewPath
% Position at one corner
   x y SetPosition
% Draw lines to other corner
   width 0 LineToRelative
   0 height LineToRelative
% Add final two sides of rectangle
   width Negate 0 LineToRelative
   ClosePathSegment
% Intersect Rectangle with CurrentClippingRegion
% yielding a new CurrentClippingRegion
   ClipPath
% Erase all trace of the rectangle path
   Newpath
}
```

31 Filters

It is often useful to perform specialized data transforms, especially decompression, on the data used as a source for raster graphics imaging operators. In SPDL these types of data transforms are provided in the form of *filters*. An important feature of filters is the ability to layer, or *pipeline*, them. Connectivity between pipelined filters is provided in the form of *StreamObjects*. This clause describes the semantics of filters, *StreamObjects*, the standard filters specified by this International Standard, and the **Filter** operator which constructs new *StreamObjects* and pipelines filters.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642753 936

31.1 Model for Filters

Filters are employed in a document via the **Filter** operator. This operator takes as operands a data source (see the description of data sources in 31.1.1; this is *not* a Data Source Resource), parameters for the filter to be used (if necessary), and the name of the filter. The **Filter** operator returns a filtered *StreamObject* that can be used as an operand to subsequent executions of the **Filter** operator or to other operators (see clause 28).

31.1.1 Data Sources for Filters

When a filtered StreamObject is accessed for data by an operator, the first data source in the filter pipeline with which the StreamObject is associated is accessed for data. The data flows through all the filters in the pipeline in turn, and the resultant filtered data is provided to the operator accessing the StreamObject.

There are three possible types of data sources for filters — StreamObjects, Procedures, and *OctetStrings*. The following subclauses describe how each is used as a filter data source

31.1.1.1 StreamObjects

It is expected that the most common data source for filters in documents will be StreamObjects. A StreamObject can be obtained in one of two ways:

— The **FindResource** operator can find a Data Source resource declared in structure and return a StreamObject associated with that Data Source.

— A StreamObject created by a previous execution of the **Filter** operator can be employed as an operand to a subsequent execution of the **Filter** operator.

The pipelining of filters is possible because a StreamObject returned by the **Filter** operator, which is the output from the data transform defined by the filter, can in turn be used as the StreamObject operand to a subsequent execution of the **Filter** operator; thus the output from one filter can be the input to another filter.

31.1.1.2 Procedures

The data source for a filter can be a Procedure. When the filter requires more data to transform, it calls the Procedure. The Procedure shall return (on the operand stack) an value of type *OctetStringReference* which references an OctetString containing an arbitrary number of octets of data. The filter then pops this OctetStringReference from the stack and uses its contents as input to the filter.

This process repeats until the filter encounters its End-of-Data condition (see 31.1.2); any leftover data in the final OctetString is discarded. The Procedure can return an OctetStringReference to an OctetString of length 0 to indicate that no more data is available.

The implementation of a filter shall use or safely store all octets in the data source OctetString to which a reference is returned by the Procedure before again calling the Procedure for more data. This allows the Procedure to allocate a single OctetString as a buffer and return the same OctetString each time with different contents.

31.1.1.3 OctetStrings

If the data source for a filter is an OctetString, the filter simply uses its contents as data to be decoded. If the filter encounters its End-of-Data condition (see 31.1.2), it ignores the remainder of the OctetString; otherwise, it continues until it has exhausted the contents of the OctetString.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642754 872

31.1.2 End-of-Data

A filter can reach a state in which it cannot continue filtering data; this is called the End-Of-Data condition (EOD). Most filters are able to detect an EOD marker encoded in the data that they are reading; the nature of this marker depends on the filter In a few instances, the EOD condition is based on predetermined information, such as an octet count or a scan line count, instead of by an explicit marker in the encoded data. The EOD condition for each filter is specified in the description of the filter in 31.2.

31.1.3 Filter Identifiers

Unlike ColorSpaceObjects, which are parameterized in structure, Filters are parameterized in content at the time of their instantiation. Filters are parameterized under program control via execution of the **Filter** operator. This allows previously defined Filter resources to be reused many times in content with different sets of parameters A Filter Identifier is obtained via execution of the **FindResource** operator with an operand that is the value of an INTERNAL RESOURCE IDENTIFIER which was previously bound to a Filter resource via a RESOURCE DECLARATION. The result returned by the **FindResource** operator is an OctetStringReference which references an OctetString which contains the canonical character string of a public identifier (as defined in ISO/IEC 9070) for the Filter. For the Filters defined in this International Standard, the owner name is "ISO/IEC 10180" and the Object Name is specified in the description of the filter in 31 2 below

31.2 Standard Filters

The following subclauses describe the semantics, names, and parameters of the standard filters defined by this International Standard. The algorithms for some filters have been formally standardized (for example, CCITTFAXDecode), while the algorithms for others are based on de facto standards in wide use in the computing industry (for example, ASCII85Decode). Conforming implementations may provide additional filters not defined by this International Standard.

31.2.1 ASCIIHexDecode

The Object Name of the ASCIIHexDecode filter is *Filter::ASCIIHexDecode*.The ASCIIHexDecode filter requires no parameters. This filter decodes ASCII-encoded pairs of octets and produces binary data.
For each pair of ASCII-encoded hexadecimal digits (0-9 and A-F or a-f), it interprets the first digit as a hexadecimal value for the most significant four bits of an octet, and the second digit as the hexadecimal value for the least significant four bits of the octet, thus it produces one octet of output data for each two octets of input. All white space characters (space, tab, carriage return, line feed, form feed, and null) are ignored. The ASCII character > indicates EOD Any other characters will cause **RaiseError** to be invoked with *DataError* as its operand

If the filter encounters EOD when it has read an odd number of hexadecimal digits, it will behave as if it had read an additional zero digit.

31.2.2 ASCII85Decode

The Object Name of the ASCII85Decode filter is *Filter::ASCII85Decode* The ASCII85Decode filter requires no parameters. This filter decodes data encoded in the ASCII base-85 encoding and produces binary data — four octets of binary data are produced for every five octets of input data. The input data is encoded as follows:

Binary data octets are encoded in 4-tuples (groups of 4) Each 4-tuple is used to produce a 5-tuple of ASCII-encoded characters. If the binary 4-tuple is (b_1 b_2 b_3 b_4) and the encoded 5-tuple is (c_1 c_2 c_3 c_4 c_5), then the relation between them is

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642755 709

$$(b_1 \times 256^3) + (b_2 \times 256^2) + (b_3 \times 256) + b_4 =$$

$$(c_1 \times 85^4) + (c_2 \times 85^3) + (c_3 \times 85^2) + (c_4 \times 85) + c_5$$

In other words, four octets of binary data are interpreted as a base-256 number and then converted into a base-85 number. The five *digits* of this number, $(c_1\ c_2\ c_3\ c_4\ c_5)$, are then converted into ASCII characters by adding 33, which is the ASCII code for !, to each. Thus, ASCII characters in the range ! to *u* are used, where ! represents the value 0 and *u* represents the value 84. As a special case, if all five digits are zero, they are represented by a single character *z* instead of by !!!!!.

All white space characters (space, tab, carriage return, line feed, form feed, and null) are ignored. The character sequence </ shall not occur in the input data. If it would normally occur in the encoding, a white space character shall be inserted between the < and the /.

If the filter encounters the character ° in its input, the next character shall be > and the filter will reach EOD. Any other characters will cause **RaiseError** to be invoked with *DataError* as its operand. Also, any character sequences that represent impossible combinations in the ASCII base-85 encoding will cause **RaiseError** to be invoked with *IOError* as its operand.

31.2.3 LZWDecode

The Object Name *Filter::LZWDecode* is reserved for the LZWDecode Filter.

NOTE 167 The LZWDecode filter requires no parameters. The filter supports the decoding scheme specified in clause 13 of the TIFF 6.0 Specification, dated June 3, 1992 available from Adobe Corporation, Seattle, Washington. The encoded data format and the decoding algorithm are as specified in that Section. The code 257 indicates EOD.

31.2.4 RunLengthDecode

The Object Name of the RunLengthDecode filter is *Filter::RunLengthDecode*. The RunLengthDecode filter requires no parameters. This filter decodes data in run-length encoded format The encoded data consists of a sequence of *runs*, where each run consists of a length octet followed by between 1 and 128 octets of data. If the length octet is in the range 0 to 127, the following *length* + *1* octets (between 1 and 128 octets) are to be copied literally to the output of the filter. If *length* is in the range 129 to 255, the following single octet is to be replicated *257* - *length* times (between 2 and 128 times) to the filter output. A run length of 128 indicates EOD.

31.2.5 CCITTFaxDecode

The Object Name of the CCITTFaxDecode filter is *Filter::CCITTFaxDecode*. The CCITTFaxDecode filter requires several parameters which are presented to the **Filter** operator in the form of a Dictionary. The filter supports both Group 3 and Group 4 encoding schemes, as specified in Recommendations T.4 and T.6 of the CCITT *Blue Book*, volume VII.3, 1988. The encoded data format and the decoding algorithm are specified in these Recommendations. If the filter encounters improperly encoded source data, **RaiseError** shall be invoked with *DataError* as its operand; the filter does not perform any error correction or re synchronization. The form in which EOD is indicated is determined by the value of the Rows key in the Dictionary

NOTE 168 SPDL support for Recommendations T.4 and T.6 is limited to the decoding of data. It does not include initial connection and handshaking protocols that would be required to communicate with an actual facsimile machine. The purpose of this filter is to provide for efficient inclusion of facsimile-ready bi-level sampled raster graphics images in SPDL documents.

The parameters in the Dictionary referenced by the *DictionaryReference* operand to the **Filter** operator for the CCITTFaxDecode filter are described in the following subclauses.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

31.2.5.1 K

The optional key/value pair <*K: Integer*> specifies the encoding scheme that was used to encode the source data. A negative value indicates pure two-dimensional (Group 4) encoding. Zero indicates pure one-dimensional encoding (Group 3, 1-D). A positive value indicates mixed one- and two-dimensional encoding (Group 3, 2-D). The default value used by the filter if the key *K* is not in the Dictionary is 0.

31.2.5.2 EndOfLine

The optional key/value pair <*EndOfLine: Boolean*> specifies whether the source data was encoded with an end-of-line bit pattern prefixed to each line of data. The default value used by the filter if the key *EndOfLine* is not in the Dictionary is false.

31.2.5.3 *EncodedByteAlign*

The optional key/value pair <*EncodedByteAlign: Boolean*> specifies whether the source data was encoded with extra zero bits prefixed to each line of data as necessary to ensure that each line begins on an octet boundary. If the value of *EncodedByteAlign* is true, the extra zero bits are present in the source data. The default value used by the filter if the key *EndOfLine* is not in the Dictionary is false.

31.2.5.4 Columns

The optional key/value pair <*Columns: Cardinal*> specifies the width of the raster graphics image data in pixels The default value used by the filter if the key *Columns* is not in the Dictionary is 1728.

31.2.5.5 Rows

The optional key/value pair <*Rows: Cardinal*> specifies the height of the raster graphics image data in scan lines. If this parameter is zero, the height of the raster graphics image data is not predetermined; the encoded data shall be terminated by an end-of-block bit pattern or by an EOD condition on the filter's data source. The default value used by the filter if the key *Rows* is not in the Dictionary is 0.

31.2.5.6 EndOfBlock

The optional key/value pair <*EndOfBlock: Boolean*> specifies whether the source data was encoded with an end-of-block bit pattern appended to the encoded data. An *EndOfBlock* value of true overrides the *Rows* parameter. The default value used by the filter if the key *EndOfBlock* is not in the Dictionary is true.

The end-of-block pattern is the CCITT end-of-facsimile-block (EOFB) or return-to-control (RTC) appropriate for the *K* parameter.

31.2.5.7 BlackIs1

The optional key/value pair <*BlackIs1: Boolean*>, if true, causes bits whose value is 1 to be interpreted as black pixels and bits whose value is 0 as white pixels. The default value used by the filter if the key *BlackIs1* is not in the Dictionary is false.

31.2.6 NullDecode

The Object Name of the NullDecode filter is *Filter::NullDecode*. The NullDecode filter performs no data transformation — the output of the filter is identical to its input. This filter enables an arbitrary data source (Procedure or OctetString) to be easily treated as a data source because it does not require the detection of an EOD marker.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The *NullDecode* filter requires two parameters

<EODstring: OctetStringReference>

<EODcount: Cardinal>

which specify the condition under which the filter is to recognize EOD. If the value of *EODcount* is positive (and the length of the *EODstring* is non-zero), the filter will allow data to pass through the filter until it has encountered exactly *EODcount* instances of the *EODstring*; then it will reach EOD. All occurrences of the *EODstring* are also passed through.

If *EODcount* is zero (and the length of the *EODstring* is non-zero), then the first occurrence of the *EODstring* results in EOD. In this case, the *EODstring* is consumed by the filter, but it will not be passed through the filter.

In each case, overlapping instances of the *EODstring* are not recognized; for example, an *EODstring* of *eee* is recognized only once in the input *XeeeeX*.

The *EODstring* may also be of length zero. In this case, if the value of *EODcount* is positive, the filter will simply pass *EODcount* octets of arbitrary data through to its output. If the value of *EODcount* is zero, detection of EOD markers is disabled; the filter will never reach EOD. This is useful primarily when using Procedures or OctetStrings as data sources.

31.3 Operators

The specification of the **Filter** operator and its semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operator. Content exceptions and exception handling are defined in clause 36. In addition to these operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6.2.

31.3.1 Filter

The **Filter** operator takes 2 or more operands

<filteridentifier: OctetStringReference>

<$param_n$: Cardinal or DictionaryReference or OctetStringReference>

<$param_1$: Cardinal or *DictionaryReference or OctetStringReference >*

<$param_0$: Cardinal or *DictionaryReference or OctetStringReference >*

<datasource: StreamObject or Procedure or OctetStringReference>

and returns one result

<stream: StreamObject>

where *filteridentifier* is a reference to an OctetString that contains the canonical character string form of the Public Object Identifier which identifies the filter and said reference is obtained via execution of the **FindResource** operator, *datasource* is the source for encoded input data appropriate to *filteridentifier*, and the number of and nature of the intervening parameters, if any, is dependent on *filteridentifier*. The result is a newly created StreamObject that is filtered according to the value of *filteridentifier* and the other parameters to the **Filter** operator. The standard filters defined by this International Standard and their parameters, if any, are defined in 31.2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642758 418

If *filteridentifier* does not identify a filter known to the implementation, **RaiseError** shall be invoked with *UndefinedKey* as its operand. If *filteridentifier* is not a reference obtained via execution of the **FindResource** operator, **RaiseError** shall be invoked with *UndefinedResource* as its operand.

32 Patterns

When operators such as **FillPath**, **StrokePath**, and **ShowString** paint areas of the page with the current color, they ordinarily apply a single color that covers the areas uniformly. Sometimes it is desirable to apply a logical ink that consists of a repeating figure instead of a simple color. Such a repeating figure is called a *pattern*. This clause describes the creation and use of patterns.

32.1 Creation and Use of Patterns

Painting with a pattern means replicating, or *tiling*, a small graphical figure (called a *pattern cell*) at fixed intervals in x and y so as to cover the areas to be painted. The appearance of a pattern cell is defined by an arbitrary Procedure; it can include graphical elements such as filled areas, text, and sampled raster graphics images. The shape of a pattern cell need not be rectangular, and the spacing of tiles can differ from the size of the pattern cell. Patterns can be used to create various graphical textures, such as weaves, brick walls, and similar geometrical tilings.

NOTE 169 Although the document normally does not have access to or knowledge of the device pixel space on any given implementation, the description of the tiling mechanism for patterns, and the related controls available to the document, requires that this clause make reference to a device pixel space.

Painting with a pattern is a four-step Procedure:

— *Describe the pattern.* This is done by creating a special-use Dictionary with prescribed contents, called a *Pattern Dictionary*, that contains information about the pattern. A crucial key/value pair in the Dictionary is the *PaintProc*, a Procedure that can be executed to paint a single pattern cell. This Dictionary is constructed via a RESOURCE DEFINITION, and is made available to content via the **FindResource** operator.

— *Instantiate the pattern.* The **MakePattern** operator copies a Pattern Dictionary and produces an instance of the pattern (in the form of a readonly Pattern Dictionary) that is locked to the RCS. In other words, the size of a pattern cell and the phase of the tiling in relation to device pixels are determined by the *CurrentTransformation* at the time **MakePattern** is executed. The pattern itself is unaffected by subsequent changes to the Graphics State Variables. All subsequent executions of the readonly Pattern Dictionary's *PaintProc* will use the same values for the Graphics State Variables as existed at the time the **MakePattern** operator was executed.

— *Select the pattern as the current color.* The operator **SetPatternColor** installs a pattern as the current color.

— *Invoke imaging operators, such as* **FillPath, StrokePath, MaskBitMap,** *or* **ShowString**. All areas that normally would be painted with a uniform color are instead tiled with the pattern cell. To accomplish this, the Pattern Dictionary's *PaintProc* is executed to obtain the pattern cell; which is then tiled on the current page as many times as necessary. It is expected that some implementations will optimize execution by maintaining a cache of recently used pattern cells.

32.2 Pattern Dictionaries

The Pattern Dictionary defines

— the type of pattern being created,

— the Procedure which paints the pattern cell,

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642759 354

— the clip for the pattern cell, and

— controls for tiling the pattern cell.

The pattern cell is described in its own coordinate system, the *pattern coordinate system*, which is defined by concatenating the *Transformation* operand of the **MakePattern** operator with the *CurrentTransformation* at the time **MakePattern** is executed. The tiling and clipping parameters are interpreted in the pattern coordinate system; the Procedure which paints the pattern cell is executed in that coordinate system.

The following subclauses describe the mandatory key/value pairs in a Pattern Dictionary. The Pattern Dictionary can also contain other (constant) information that is required by the *PaintProc*. All key/value pairs except *Implementation* can appear in a Pattern Dictionary (a reference to which is an operand to the **MakePattern** operator).

32.2.1 PaintType

The mandatory key/value pair *<PaintType: Cardinal>* determines how the color of the pattern cell is to be specified. The valid values for *PaintType* are shown in table 36.

Table 36

Value	PaintType
1	*Colored pattern PaintProc* itself specifies the color space(s) and color(s) used to paint the pattern cell. The *PaintProc* must select a color space before performing any imaging operators
2	*Mask pattern PaintProc* does not specify any color space or color information; instead, the entire pattern cell is painted with a separately specified color each time the pattern is used. In effect, the *PaintProc* describes a mask through which the current color is to be poured. The *PaintProc* must not execute any of the color operators (this includes all operators in clause 34), the **SetTransfer** operator, the **SetPatternColor** operator, or the **ImageRasterElement** operator. Use of the **MaskBitMap** operator is permitted, since it does not specify any color information

32.2.2 PaintProc

The mandatory key/value pair *<PaintProc: Procedure>* specifies the Procedure used to paint a pattern cell. *PaintProc* takes one operand

<PatternDict: DictionaryReference>

where *PatternDict* references the Pattern Dictionary containing *PaintProc*. *PaintProc* returns no results. The effect of executing *PaintProc* is to draw one pattern cell.

The Graphics State in which *PaintProc* executes is the Graphics State at the time when the pattern was instantiated via execution of the **MakePattern** operator, including the *CurrentTransformation* at that time as modified by concatenating it with the *Transformation* operand to the **MakePattern** operator.

If *PaintProc* executes the **ImageRasterElement** or **MaskBitMap** operators, none of the underlying Data Sources for the operation may be *DataSource::Document*. If any of the underlying Data Sources is a StreamObject derived from an external data source, the execution of the **FindResource** operator to obtain the StreamObject must occur within *PaintProc*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642760 076

NOTE 170 Because it is expected that some implementations will optimize execution by maintaining a cache of recently used pattern cells, *PaintProc* may be executed at unpredictable times and in unpredictable contexts. Therefore *PaintProc* should rely only on the operators in *SystemDict* and on the contents of its *PatternDict* operand, and should always produce the same results For further restrictions on the operators which may be performed by *PaintProc*, see the description of *PaintType* in 32 2.1.

32.2.3 BBox

The mandatory key/value pair *<BBox: VectorReference>* is a reference to a 4-element Vector, each element of which is of type *Number*, specifying the bounding box of the pattern cell, measured in the pattern cell coordinate system This bounding box is used to clip the pattern cell. The four elements of the Vector specify the lower left x, lower left y, upper right x, and upper right y coordinates of the bounding box:

[llx lly urx ury]

32.2.4 XStep

The mandatory key/value pair *<XStep: Number>* specifies the desired horizontal spacing between pattern cells, measured in the pattern cell coordinate system.

NOTE 171 The values of *XStep* and *YStep* can differ from the dimensions of the pattern cell implied by the *BBox* key/value pair; this permits tiling with irregularly-shaped figures.

32.2.5 YStep

The mandatory key/value pair *<YStep: Number>*specifies the desired vertical spacing between pattern cells, measured in the pattern cell coordinate system

NOTE 172 The values of *XStep* and *YStep* can differ from the dimensions of the pattern cell implied by the *BBox* key/value pair, this permits tiling with irregularly-shaped figures.

32.2.6 Tiling

The placement of pattern cells in the tiling is based on the location of one key pattern cell, which is then displaced by multiples of *XStep* and *YStep* in order to replicate the pattern cell. The origin of the key pattern cell coincides with the origin of the coordinate system defined by concatenating the *Transformation* operand of the **MakePattern** operator with the *CurrentTransformation* at the time **MakePattern** is executed. Thus, the phase of the tiling can be controlled by the translation components of the *Transformation* operand. This tiling is frozen; that is, whenever the Pattern Dictionary created by **MakePattern** is used for painting, the same tiling is used, regardless of intervening changes to the *CurrentTransformation*

32.2.7 TilingType

The mandatory key/value pair *<TilingType: Cardinal>* controls adjustments to this tiling in order to quantize it to the device pixel grid. The valid values for *TilingType* are shown in table 37

Table 37

Value	TilingType
1	*Constant spacing* consistently, that is, by a multiple of a device pixel. To achieve this, the **MakePattern** operator may need to distort the pattern slightly, by making small adjustments to *XStep* , *YStep* , and the *Transformation* operand to the **MakePattern** operator. The amount of distortion shall not exceed one device pixel.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Value	TilingType
2	*No distortion* but the spacing between pattern cells may vary by as much as one device pixel in both x and y dimensions when the pattern is painted. This achieves the spacing requested by *XStep* and *YStep* on average, but not for individual pattern cells.
3	*Constant spacing and faster tiling* Like *TilingType* 1, but with additional distortion of the pattern cell permitted in order to enable a more efficient implementation. The amount of additional distortion allowed is implementation-specific.

32.2.8 Implementation

The key/value pair *<Implementation: Any>* is added to the Pattern Dictionary by the **MakePattern** operator. It is intended to be used by implementations to store information to aid in proper tiling and possibly caching of the pattern. The type and value of this key/value pair are implementation-dependent. If this key is present in a Pattern Dictionary presented to the **MakePattern** operator, its value may be overwritten by the **MakePattern** operator.

32.3 Pattern Semantics

The number of and nature of the operands required by the **SetPatternColor** operator depends on the type of pattern (colored or mask) being used. The semantics of the **SetPatternColor** operator for each pattern type are specified in 32.3.1 and 32.3.2, respectively.

32.3.1 Colored Patterns

A colored pattern is one whose color is self-contained. As part of painting the pattern cell, the *PaintProc* explicitly sets the colors of all graphical elements that it paints. A single pattern cell can contain elements that are painted different colors; it can also contain sampled gray-scale or color images.

The operands required by the **SetPatternColor** operator to select a colored pattern are

<PatternDict: DictionaryReference>

where *PatternDict* references a Pattern Dictionary obtained via execution of the **MakePattern** operator on a Pattern Dictionary whose *PaintType* is 1.

32.3.2 Mask Patterns

A mask pattern is one that does not have any inherent color; instead, the color must be specified separately whenever the pattern is used. This provides a way to tile different regions of the page with pattern cells having the same shape but different colors. The pattern's *PaintProc* does not explicitly specify any colors; it cannot contain sampled gray-scale or color images (but it may use the **MaskBitMap** operator).

The operands required by the **SetPatternColor** operator to select a mask pattern, and a color with which to paint the mask pattern, are

<PatternDict: DictionaryReference>

<comp$_n$: Any>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642762 949

<$comp_1$. *Any*>

where *PatternDict* references a Pattern Dictionary obtained via execution of the **MakePattern** operator whose *PaintType* is 2, and the remaining operands are the appropriate components to select the color to be used to paint the pattern in the underlying color space. The *underlying* color space is the color space which is the value of the *CurrentColorSpace* Graphics State Variable (as specified by the most recent execution of the **SetColorSpace** operator in the current context of execution).

NOTE 173 Executing the **SetColorSpace** operator after the **SetPatternColor** operator both changes *CurrentColorSpace* and initializes *CurrentColor* removing the pattern that was current.

32.4 Obtaining a Prototype Pattern Dictionary for Use in Content

A reference to a Pattern Dictionary to be used as an operand to the **MakePattern** operator is obtained via execution of the **FindResource** operator. The **FindResource** operator takes as an operand the value of an INTERNAL RESOURCE IDENTIFIER that has been previously bound to a Pattern resource in structure by a RESOURCE DECLARATION, and returns a reference to a Pattern Dictionary .

NOTE 174 The **FindResource** operator does not examine the Pattern Dictionary to ensure that the required key/value pairs are present and of the correct type; this function is performed by the **MakePattern** operator.

32.5 Operators

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and exception handling are defined in clause 36. In addition to these operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6 2.

32.5.1 MakePattern

The **MakePattern** operator takes two operands

<*T: Transformation*>

<*PatternDictionary: DictionaryReference*>

and returns one result

<*NewPatternDict: DictionaryReference*>

where *PatternDictionary* is obtained via execution of the **FindResource** operator. If *PatternDictionary* is not a reference to a Dictionary obtained via execution of the **FindResource** operator, **RaiseError** shall be invoked with *UndefinedResource* as its operand. The set of mandatory key/value pairs is checked. If any keys are missing, **RaiseError** shall be invoked with *UndefinedKey* as its operand If the key is present but the value bound to it is of the wrong type, **RaiseError** shall be invoked with *TypeCheck* as its operand. If the key is present and the type is correct but the bound value is out of range, **RaiseError** shall be invoked with *RangeCheck* as its operand.

The *pattern coordinate system* is constructed from the current Graphics State as follows:

- The operand *T* is concatenated with a copy of the *CurrentTransformation* in the current Graphics State. The resulting *Transformation* is then adjusted to ensure that the device pixel space can be tiled properly with a pattern cell of the given size in accordance with the *TilingType* specified in *PatternDictionary*.
- The *CurrentPath* is set to the null path, as if by execution of the **NewPath** operator

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642763 885

— The *CurrentClippingRegion* is replaced by the pattern cell bounding box (defined by the *BBox* key/value pair in *PatternDictionary*).

NOTE 175 This description is intended to specify the semantics of the **MakePattern** operator only; it is not intended to specify how implementations satisfy the semantics for patterns.

MakePattern then makes a new copy of *PatternDictionary*, adds the *Implementation* key/value pair, reduces the access attribute of the Dictionary to at least *ReadOnly*, and pushes it onto the Operand Stack.

NOTE 176 Because on some implementations the **MakePattern** operator may further alter the contents of the Dictionary, the reduction of the access attribute of the Dictionary is performed to prevent subsequent content from modifying the contents of the Dictionary. Implementations which do not wish to allow any subsequent access to the contents of the Dictionary (except via executions of **SetPatternColor**) may reduce the access attribute of the Dictionary to *ExecuteOnly*.

32.5.2 SetPatternColor

The nature and number of operands required by the **SetPatternColor** operator depend upon the type of pattern (colored or mask) being set; these operands are described in 32.3.1 and 32.3.2 . The **SetPatternColor** operator returns no results. The effect of executing the **SetPatternColor** operator is to set the value of the *CurrentColor* Graphics State Variable as described in 34.6.4.

If any of the required key/value pairs are missing from the Pattern Dictionary operand of the **SetPatternColor** operator, **RaiseError** shall be invoked with *UndefinedKey* as its operand.

33 Forms

A *form* is a self-contained description of any arbitrary geometric graphics, text, or raster graphics that are to be imaged multiple times, either on one page or on several pages.

33.1 Model for Forms

A form is defined at the structure level via a RESOURCE DEFINITION, and is bound to the value of an INTERNAL RESOURCE IDENTIFIER in a RESOURCE DECLARATION. The appearance of a form is described by a Procedure that executes imaging operators. The **FindResource** operator is used in content to obtain a form, which is represented in content by a *Form Object*, and is imaged by providing the Form Object as an operand to the **ExecuteForm** operator.

NOTE 177 There is no *Form Object* type in SPDL; rather the term is used as a convenient term to describe both references to the particular objects of type *Dictionary* which represent a form and its associated parameters (and are obtained via the **FindResource** operator) and to the Dictionaries themselves. That is, Form Object will be used both for the *DictionaryReference* and the Dictionary; which is intended will be clear from context.

The remainder of this clause specifies the contents of Form Dictionaries and the semantics of the **ExecuteForm** operator.

33.2 Form Dictionaries

A form is represented by a Form Object, which is a reference to a Form Dictionary with prescribed contents which fully describes the form to be imaged. The form is defined in its own coordinate system, the *form coordinate system*, which is defined by concatenating the value bound to the *Matrix* key in the Form Dictionary with the *CurrentTransformation* each time the Form Objects's *PaintProc* (the Procedure in the Form Dictionary that images the form) is executed. The *BBox* parameter in the Form Dictionary is interpreted in the form coordinate system; the *PaintProc* is executed in that coordinate system.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642764 711

The following subclauses describe the contents of a Form Dictionary. In addition to the contents described below, the Form Dictionary can also contain other (constant) information that is required by the *PaintProc* (see 33.2.3)

33.2.1 BBox

The mandatory key/value pair *<BBox: Vector>* is a 4-element Vector of type *Number* specifying the bounding box of the form in the form coordinate system. This bounding box is used to clip the form. The four elements of the Vector specify the lower left x, lower left y, upper right x, and upper right y coordinates of the bounding box:

[llx lly urx ury]

33.2.2 Matrix

The mandatory key/value pair *<Matrix: Transformation>* is the Transformation to be concatenated with the *CurrentTransformation* to produce the Transformation that defines the form coordinate system.

33.2.3 PaintProc

The mandatory key/value pair *<PaintProc: Procedure>* specifies the Procedure used to paint the form. *PaintProc* takes one operand

<FormDict: DictionaryReference>

where *FormDict* is a reference to the Form Dictionary containing *PaintProc*, and returns no results. The effect of executing *PaintProc* is to image one instance of the form. The Context of Interpretation for *PaintProc* is described in 33.4

If *PaintProc* executes the **ImageRasterElement** or **MaskBitMap** imaging operators, none of the underlying Data Sources for the operator may be *DataSource::Document*. If any of the underlying Data Sources is a *StreamObject* derived from an external data source, the execution of the **FindResource** operator to obtain the *StreamObject* shall occur within *PaintProc*.

NOTE 178 Because it is expected that many implementations will optimize execution by maintaining a cache of recently used forms, *PaintProc* may be executed at unpredictable times and in unpredictable contexts. Therefore *PaintProc* should rely only on the operators in *SystemDict*, on the contents of its *FormDict* operand, and the Graphics State Variables not set to their default values according to 33.4. It should always produce the same results. It should have no side effects besides painting the form

33.2.4 UniqueID

The optional key/value pair *<UniqueID: Identifier>* is designed to uniquely identify, for caching purposes, a form that is to be used by multiple documents.

NOTE 179 To avoid the possible waste of memory resources in caching implementations, a Form Dictionary for a form that is specified and used within a single document should not contain this key.

33.2.5 Implementation

The key/value pair *<Implementation: Any>* is added to the Form Dictionary by the **ExecuteForm** operator. It is intended to be used by implementations to store information to aid in the caching of the form. The type and value of this key are implementation-dependent. If an key/value pair whose key is *Implementation* is placed in a Form Dictionary by the TOKENSEQUENCE that defines the Dictionary, its value may be overwritten by the **ExecuteForm** operator.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642765 658

33.3 Obtaining a Form Dictionary for Use in Content

A Form Object to be used as an operand to the **ExecuteForm** operator is obtained via execution of the **FindResource** operator. The **FindResource** operator takes as an operand an INTERNAL RESOURCE IDENTIFIER that has been previously bound to a Form resource in structure by a RESOURCE DECLARATION, and returns a Form Object.

NOTE 180 The **FindResource** operator does not examine the Form Dictionary referenced by the Form Object to ensure that the required keys are present and of the correct type; this function is performed by the **ExecuteForm** operator.

33.4 Imaging a Form

A form is imaged via execution of the **ExecuteForm** operator, which in turn executes the *PaintProc* in the Form Objects's Form Dictionary. Prior to executing the **ExecuteForm** operator, a document should set appropriate parameters in the graphics state; in particular, it should alter the *CurrentTransformation* in order to control the position, size, and orientation of the form in User Coordinate Space.

Before execution of the *PaintProc*, the following Graphics State Variables are reset to their default values (see clause 19 for the default value for each):

— *CurrentDashPattern*

— *CurrentMiterLimit*

— *CurrentPath*

— *CurrentPosition*

— *CurrentStrokeEnd*

— *CurrentStrokeJoin*

— *CurrentStrokeWidth*

NOTE 181 If some other value than the default value of the above Graphics State Variables is desired, additional key/value pairs may be defined in the Form Dictionary to represent Graphics State Variable values or other information that can be used by the *PaintProc* in imaging the form.

The rest of the Virtual Machine State in which the *PaintProc* executes is implementation-dependent.

NOTE 182 In some implementations, part or all of the remainder of the Virtual Machine State may be inherited from the Virtual Machine State at the time of execution of the **ExecuteForm** operator.

33.5 ExecuteForm

The specification of the **ExecuteForm** operator and its semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operator. Content exceptions and exception handling are defined in clause 36. In addition to the operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36 6 2.

The **ExecuteForm** operator accepts one operand

<FormObject: DictionaryReference>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642766 594

where *Form Object* is obtained via execution of the **FindResource** operator. The operator returns no results. The graphical output produced by **ExecuteForm** is defined by the *PaintProc* Procedure (see 33.2.3) in the Form Object's Form Dictionary. If this is the first invocation of **ExecuteForm** for the form, **ExecuteForm** first verifies that the Form Dictionary contains the required keys. It then adds the *Implementation* key/value pair (see 33.2 5) and reduces the access attribute of the Dictionary to at least *ReadOnly*.

NOTE 183 Because on some implementations the **ExecuteForm** operator may alter the contents of the Dictionary, the reduction of the access attribute of the Dictionary is performed to prevent subsequent content operators from modifying the contents of the Dictionary. Implementations which do not wish to allow any subsequent access to the contents of the Dictionary (except via executions of **ExecuteForm**) may reduce the access attribute of the Dictionary to *ExecuteOnly*.

If any of the required key/value pairs are missing from the Form Dictionary operand of the **ExecuteForm** operator, **RaiseError** shall be invoked with *UndefinedKey* as its operand.

ExecuteForm may call the *PaintProc* to image the form. It shall call the *PaintProc* the first time that **ExecuteForm** is called with a given Form Object and whenever changes to the Graphics State since the last time the *PaintProc* was called would effect how the form was imaged Changes to the translation components of the *CurrentTransformation* Graphics State Variable do not effect how the form will be imaged. When ExecuteForm needs to call the *PaintProc* Procedure (rather than using a cached representation to image the form), it pushes the Form Object on the operand stack, then performs the equivalent of the following:

```
    SaveGraphicsState
% The following procedure resets the Graphics State Variables specified in 33.4 to their default values;
% the code for this Procedure is obvious. The values of all other Graphics State Variables are taken
% from the inherited Graphics State.
    SetGStateVariables
% Concatenate Form Dictionary's Matrix with CurrentTransformation
    Dup /Matrix Get Concat
% Stack has. Form-Dictionary llx lly urx ury (for BBox)
    Dup /BBox Get VectorLoad Pop
% Stack has: Form-Dictionary llx lly width height
    Exchange 3 Index Subtract
    Exchange 2 Index Subtract
% Clip to form bounding box; ends with NewPath
    RectClip
% Execute PaintProc
    Dup /PaintProc Get Execute
    RestoreGraphicsState
```

NOTE 184 The *PaintProc* Procedure is expected to consume the DictionaryReference operand and to execute a sequence of imaging operators to paint the form.The execution of *PaintProc* should depend only on the contents of the Form Dictionary and UserDict plus the Graphics State Variables not set to their default values according to 33.4,. The *PaintProc* shall always produce the same output, given the same values for the Graphics State Variables not set to their default values according to 33.4, independent of the number of times it is called and independent, for example, of the contents of *UserDict*. The document should not expect any particular execution of **ExecuteForm** to cause execution of the specified *PaintProc*.

If *Form Object* is not a reference to a Dictionary obtained via execution of the **FindResource** operator, **RaiseError** shall be invoked with *UndefinedResource* as its operand.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642767 420

34 Color Space and Color Operators

34.1 Model for Color Spaces and Colors

A *color space* is a coordinate system in terms of which a given color may be specified. A wide variety of color spaces are in common use. This clause defines the method by which documents select color spaces within which to specify colors, and the method by which a given color is then specified within a selected color space.

Colors are closely associated with the notion of *ink* described in 17.2.1 . Imaging operators cause the currently selected color in the currently selected color space to be painted onto the page image as an *ink*..

SPDL provides a number of color spaces. An SPDL document can be assemble from PICTUREs constructed in many different color spaces. An application is not required to translate all color spaces to a single color space; instead, the application may simply describe the intended color space to SPDL using the wide collection of color spaces SPDL understands.

NOTE 185 This International Standard defines the ideal result of the presentation of an SPDL document. Conforming presentation processes need not produce this ideal result, but may instead produce a reasonable approximation subject to the limitations of the imaging technology. It is recognized that this is a subjective description.

34.2 Color Spaces and Color Space Families

Some color spaces, the device color spaces (*DeviceGray*, *DeviceRGB*, *DeviceCMYK*, *DeviceKX*), are directly related to their representation on an output device; other color spaces, the CIE and CIE-based color spaces, are related to human visual perception and are designed to be device independent. Certain special features such as *named colors* and *mappings* for indexed colors, are also modelled as color spaces, the special-purpose color spaces.

The color spaces are grouped into families of similar color spaces. Each family has been given a Public Object Identifier, the Color Space Familiy Identifier. A particular color space is identified by a combination of a Color Space Family Identifier and zero or more *parameters* that further select a particular color space within the identified family of color spaces.

SPDL documents select color spaces via execution of the **SetColorSpace** operator, which affects the contents of the *CurrentColorSpace* Graphics State Variable. The **SetColorSpace** operator takes as its operand an object reference to a Vector containing a Color Space Family Identifier and zero or more parameters appropriate to that color space family. This VectorReference is called a *Color Space Object*.

NOTE 186 There is no *Color Space Object* type in SPDL; rather the term is used as a convenient term to describe both (1) references to the particular objects of type *Vector* which are obtained via the **FindResource** operator and which represent a Color Space Family Identifier and the parameters associated with it and (2) the Vectors themselves. That is, the term Color Space Object will be used both for the VectorReference and the Vector; which is intended will be clear from context.

A Color Space Object is obtained via execution of the **FindResource** operator. This operator takes as its operand the value of an INTERNAL RESOURCE IDENTIFIER which is bound to a Color Space Object via a RESOURCE DECLARATION.

The semantics of the elements of Color Space Objects used to identify each standard color space to the **SetColorSpace** operator are defined in 34.3 through 34.5.

Once a color space has been selected, particular colors within that color space are specified via execution of the **SetColor** operator; execution of this operator affects the contents of the *CurrentColor* Graphics State Variable. The operands to the **SetColor** operator are color components which specify a specific color within the current color space; the number and valid range of these operands varies according to the current color space.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642768 367

NOTE 187 Whenever the **SetColorSpace** operator is excuted, the *CurrentColor* Graphics State Variable is initialized to a color that is defined as part of the color space that is being selected. Where possible, the default color is specified as either Black or the closest color to Black in the selected color space.

The required color components for specifying colors in each standard color space via execution of the **SetColor** operator are defined in 34.3 through 34.5.

The following subclauses define each of the standard color spaces defined by SPDL, the semantics of the parameters required by the **SetColorSpace** operator for each color space, the semantics of the color components required by the **SetColor** operator to specify a color within each color space, and the value of the *CurrentColor* Graphics State Variable (as returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator. In addition, the Object Name of the Public Object Identifier for each color space family defined in this International Standard is indicated; the Color Space Family Identifier that occurs within the Color Space Object is an OctetStringReference which references an OctetString that contains the canonical character string form of the full Public Identifier (see 9 6).

34.3 CIE and CIE-based Color Spaces

CIE color spaces enable a document to specify color in a way that is related to human visual perception. A given CIE color specification should result in uniform results (given physical device limitations) across implementations. These are device independent color spaces. The CIE color spaces are:

— *CIELAB - CIE 1976 (L*a*b*)*-space

— *CIELUV - CIE 1976 (L*u*v*)*-space

— *CIEBasedABC* - A two-stage, non-linear transformation of *CIE 1931 (XYZ)*-space

— *CIEBasedA* - An achromatic *CIEBasedABC* space

NOTE 188 The *CIEBasedABC* color space can be parameterized to form *Calibrated RGB* color spaces; see 34.3.3.

34.3.1 CIELAB

The *CIELAB* color space is equivalent to the *CIE 1976 (L*a*b*)*-space. The Object Name of the Public Object Identifier for this color space family is *color-space-family*::*CIELAB*. The Color Space Object for this color space has a single parmater.

<Dict: Dictionary>

NOTE 189 The contents of *Dict* are described in 34 3 1.1 through 34 3 1 2

The color components required by the **SetColor** operator to specify a color in the CIELAB color space are

<b: Number>

<a: Number>

<L: Number>

where the valid range for the value of *L* is 0 to 100 inclusive, and the valid ranges for the values of *a* and *b* shall be within the spectrum locus of the monochromatic stimuli.

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for this color space is

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642769 2T3

<0.0: Real>

<0.0: Real>

<0.0: Real>

The mandatory and optional key/value pairs in *Dict*, and their semantics, are described below.

34.3.1.1 WhitePoint

The mandatory key/value pair *<WhitePoint: VectorReference>* references a 3-element Vector whose elements, of type *Number*, specify the *CIE 1931 (XYZ)*-space tristimulus value of the diffuse whte point. Each element shall have value greater than 0, and the Y component shall be equal to 1. The order of elements in the Vector is:

$[\ X_w\ 1\ Z_w\ ]$

34.3.1.2 BlackPoint

The optional key/value pair *<BlackPoint: VectorReference>* references a 3-element Vector whose elements, of type *Number*, specify the *CIE 1931 (XYZ)*-space tristimulus value of the diffuse black point. Each element shall have value greater than 0, and the order of elements in the Vector is:

$[\ X_b\ Y_b\ Z_b\ ]$

If the *BlackPoint* key is not present, the default value used is:

[0 0 0]

34.3.1.3 Range

The mandatory key/value pair *<Range: VectorReference>* references a 6-element Vector whose elements, of type *Number*, specify the valid ranges of values for the L^*, a^*, and b^* components of the color space. The elements of *Range* are interpreted as three pairs of bounds, one pair for each of the three color components:

$[\ L_0\ L_1\ a_0\ a_1\ b_0\ b_1\ ]$

The valid ranges of values for each of the three color components are then defined as $L_0 \sigma L^* \sigma L_1$, $a_0 \sigma a^* \sigma a_1$, and $b_0 \sigma b^* \sigma b_1$. The values of L_0 and L_1 shall be in the range 0 to 100 inclusive.

34.3.2 CIELUV

The CIELUV color space is equivalent to the *CIE 1976 ($L^*u^*v^*$)*-space. The Object Name of the Public Object Identifier for this color space family is *color-space-family::CIELUV*. The Color Space Object for this color space has a single parameter:

<Dict: Dictionary>

NOTE 190 The contents of *Dict* are described in 34.3.2.1 through 34.3.2.2.

The color components required by the **SetColor** operator to specify a color in the CIELUV color space are

<v: Number>

<u: Number>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

<*L: Number*>

where the valid range of L is 0 to 100 inclusive, and the valid ranges for the values of u and v shall be within the spectrum locus of the monochromatic stimuli.

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for this color space is

<*0 0: Real*>

<*0.0: Real*>

<*0.0: Real*>

The mandatory and optional key/value pairs in *Dict*, and their semantics, are described below.

34.3.2.1 WhitePoint

The mandatory key/value pair <*WhitePoint: VectorReference*> references a 3-element Vector whose elements, of type *Number* specify the *CIE 1931 (XYZ)*-space tristimulus value of the diffuse white point. Each element shall have a value greater than 0, and the Y component shall be equal to 1. The order of elements in the Vector is:

$[\ X_w\ \ 1\ \ Z_w\ ]$

34.3.2.2 BlackPoint

The optional key/value pair <*BlackPoint: VectorReference*> references a 3-element Vector whose elements, of type *Number* specify the *CIE 1931 (XYZ)*-space tristimulus value of the diffuse black point. Each element shall have a value greater than 0, and the order of elements in the Vector is:

$[\ X_b\ \ Y_b\ \ Z_b\ ]$

If the *BlackPoint* key is not present, the default value used is:

[0 0 0]

34.3.2.3 Range

The mandatory key/value pair <*Range: VectorReference*>
references a 6-element Vector whose elements, of type *Number*, specify the valid ranges of values for the L^*, u^*, and v^* components of the color space. The elements of *Range* are interpreted as three pairs of bounds, one pair for each of the three color components:

$[\ L_0\ \ L_1\ \ u_0\ \ u_1\ v_0\ \ v_1\ ]$

The valid ranges of values for each of the three color components are then defined as $L_0 \sigma L^* \sigma L_1$, $u_0 \sigma u^* \sigma u_1$, and $v_0 \sigma v^* \sigma v_1$. The values of L_0 and L_1 shall be in the range 0 to 100 inclusive.

34.3.3 CIEBasedABC

The *CIEBasedABC* color space models a simple zone theory of color vision, consisting of a non-linear trichromatic first stage combined with a non-linear opponent color second stage. One common use for this transformation of the *CIE 1931 (XYZ)* -space is to ensure that low-luminance colors in sampled raster graphics images are digitized with the minimum loss of fidelity. The three components of this color space, arbitrarily

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642771 951

named *A*, *B*, and *C*, can represent any three orthogonal color components, depending on how the color space is parameterized.

NOTE 191 Among the more interesting color components which can be represented by the *A*, *B*, and *C* components of the CIEBasedABC color space are:

- *R*, *G*, and *B* in a calibrated RGB space.
- *X*, *Y*, and *Z* in the *CIE 1931 (XYZ)* -space.
- *Y*, *I*, and *Q* in the NTSC space.
- *Y*, *U*, and *V* in the SECAM and PAL spaces.

The Object Name of the Public Object Identifier for this color space family is *color-space-family::CIEbasedABC*. The Color Space Object for this color space has a single parameter:

<Dict: Dictionary>

NOTE 192 The contents of *Dict* are described in 34.3.3.1 through 34.3.3.8.

The color components required by the **SetColor** operator to specify a color in the CIEBasedABC color space are

<C: Number>

<B: Number>

<A: Number>

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for this color space is

<0.0: Real>

<0.0: Real>

<0.0: Real>

NOTE 193 If the valid range of values for a color component does not include the value 0.0, the nearest value in the valid range is substituted as the default value for that color component.

The mandatory and optional key/value pairs in *Dict*, and their semantics, are described in the subclauses below. With reference to those descriptions, the two-stage mapping from *A*, *B*, and *C* values to *CIE 1931 (XYZ)*-space is defined:

$$L = D_A(A) \times L_A + D_B(B) \times L_B + D_C(C) \times L_C$$

$$M = D_A(A) \times M_A + D_B(B) \times M_B + D_C(C) \times M_C$$

$$N = D_A(A) \times N_A + D_B(B) \times N_B + D_C(C) \times N_C$$

$$X = D_L(L) \times X_L + D_M(M) \times X_M + D_N(N) \times X_N$$

$$Y = D_L(L) \times Y_L + D_M(M) \times Y_M + D_N(N) \times Y_N$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642772 898

$$Z = D_L(L) \times Z_L + D_M(M) \times Z_M + D_N(N) \times Z_N$$

34.3.3.1 WhitePoint

The mandatory key/value pair <*WhitePoint: VectorReference*> references a 3-element Vector whose elements, of type *Number* specify the *CIE 1931 (XYZ)*-space tristimulus value of the diffuse white point. Each element shall have a value greater than 0, and the Y component shall be equal to 1. The order of elements in the Vector is:

$[\ X_w\ \ 1\ \ Z_w\ ]$

34.3.3.2 BlackPoint

The optional key/value pair <*BlackPoint: VectorReference*> references a 3-element Vector whose elements, of type *Number* specify the *CIE 1931 (XYZ)*-space tristimulus value of the diffuse black point. Each element shall have a value greater than or equal to 0, and the order of elements in the Vector is:

$[\ X_b\ \ Y_b\ \ Z_b\ ]$

If the BlackPoint key is not present, the default value used is:

[0 0 0]

34.3.3.3 RangeABC

The optional key/value pair <*RangeABC: VectorReference*> references a 6-element Vector whose elements, of type *Number*, specify the valid ranges of values for the *A*, *B*, and *C* components of the color space. The elements of *RangeABC* are interpreted as three pairs of bounds, one pair for each of the three color components:

$[\ A_0\ \ A_1\ \ B_0\ B_1\ \ C_0\ \ C_1\ ]$

The valid ranges of values for each of the three color components are then defined as

$A_0 \sigma A \sigma A_1$, $B_0 \sigma B \sigma B_1$, and $C_0 \sigma C \sigma C_1$

If the RangeABC key is not present, the default valid ranges are:

[0 1 0 1 0 1]

34.3.3.4 DecodeABC

The optional key/value pair <*DecodeABC: VectorReference*> references a 3-element Vector whose elements, of type *Procedure*, are Procedures that map *ABC* values into values which are linear with respect to the intermediate *LMN* representation. The order of the elements in the Vector is:

$[\ D_A\ D_B\ D_C\ ]$

Each Procedure, D_A, D_B, D_C, is called with the value of *A*, *B*, or *C*, respectively, as its operand and shall return the corresponding linear value.

NOTE 194 Since these Procedures are called at implementation-dependent times, they shall operate as pure functions —they should not depend on any non-default context and should not affect the current context.

If the *DecodeABC* key is not present, the default value used is a reference to a Vector of null Procedures which do not affect the contents of the operand stack:

[{} {} {}]

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

34.3.3.5 MatrixABC

The optional key/value pair *<MatrixABC: VectorReference>* references a 9-element Vector whose elements, of type *Number* specify the linear interpretation of the decoded *A*, *B*, and *C* components of the color space with respect to the intermediate *LMN* representation. The order of the elements in the Vector is:

$$[\ L_A\ \ M_A\ \ N_A\ L_B\ \ M_B\ \ N_B\ \ L_C\ M_C\ \ N_C\ ]$$

If the MatrixABC key is not present, the default matrix used is:

[1 0 0 0 1 0 0 0 1]

34.3.3.6 RangeLMN

The optional key/value pair *<RangeLMN: VectorReference>* references a 6-element Vector whose elements, of type *Number*, specify the valid ranges of values for the *L*, *M*, and *N* components of the intermediate representation. The elements of *RangeLMN* are interpreted as three pairs of bounds, one pair for each of the three color components:

$$[\ L_0\ \ L_1\ \ M_0\ M_1\ \ N_0\ \ N_1\ ]$$

The valid ranges of values for each of the three color components are then defined as

$L_0 \sigma L \sigma L_1$, $M_0 \sigma M \sigma M_1$, and $N_0 \sigma N \sigma N_1$.

If the RangeLMN key is not present, the default valid ranges are:

[0 1 0 1 0 1]

34.3.3.7 DecodeLMN

The optional key/value pair *<DecodeLMN: VectorReference>* references a 3-element Vector whose elements, of type *Procedure*, are Procedures that map *LMN* values into values which are linear with respect to *CIE 1931 (XYZ)*-space. The order of the elements in the Vector is:

$$[\ D_L\ \ D_M\ \ D_N\ ]$$

Each Procedure, D_L, D_M ,D_N, is called with the value of *L*, *M*, or *N*, respectively, as its operand and shall return the corresponding linear value.

NOTE 195 Since these Procedures are called at implementation-dependent times, they shall operate as pure functions — they should not depend on any non-default context and should not affect the current context.

If the *DecodeLMN* key is not present, the default value used is a reference to a Vector of null Procedures which do not affect the contents of the operand stack:

[{} {} {}]

34.3.3.8 MatrixLMN

The optional key/value pair *<MatrixLMN: VectorReference>* references a 9-element Vector whose elements, of type *Number* specify the linear interpretation of the decoded *L*, *M*, and *N* components of the intermediate LMN representation with respect to the *CIE 1931 (XYZ)*-space. The order of the elements in the Vector is:

$$[\ X_L\ \ Y_L\ \ Z_L\ X_M\ \ Y_M\ \ Z_M\ X_N\ \ Y_N\ \ Z_N\ ]$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642774 660

If the MatrixLMN key is not present, the default matrix used is:

[1 0 0 0 1 0 0 0 1]

34.3.4 CIEBasedA

The *CIEBasedA* color space is a one-dimensional CIEBased color space which is usually achromatic. Color values in this space have a single component, arbitrarily named *A*, which can represent a variety of color components, depending on how the *CIEBasedA* color space is parameterized.

NOTE 196 Among the more interesting color components which can be represented by the *A* component of the *CIEBasedA* color space are:

— The gray component of a calibrated gray space

— The luminance *Y* component of the *CIE 1931 (XYZ)* -space.

— The psychometric lightness L^* component of the *CIE 1976* ($L^*a^*b^*$) -space.

— The luminance *Y* component of the NTSC, SECAM, and PAL spaces.

The Object Name of the Public Object Identifier for this color space family is *color-space-family::CIEbasedA*. The Color Space Object for this color space has a single parameter.

<Dict. Dictionary>

NOTE 197 The contents of *Dict* are described in 34.3.4.1 through 34 3 4.8.

The color component required by the **SetColor** operator to specify a color in the CIEBasedA color space is

<A: Number>

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for this color space is

<0.0: Real>

NOTE 198 If the valid range of values for the color component does not include the value 0.0, the nearest value in the valid range is substituted as the default value for the color component.

The mandatory and optional key/value pairs in *Dict*, and their semantics, are described in the subclauses below. With reference to those descriptions, the two-stage mapping from *A* values to *CIE 1931 (XYZ)*-space is defined:

$$L = D_A(A) \times L_A$$

$$M = D_A(A) \times M_A$$

$$N = D_A(A) \times N_A$$

$$X = D_L(L) \times X_L + D_M(M) \times X_M + D_N(N) \times X_N$$

$$Y = D_L(L) \times Y_L + D_M(M) \times Y_M + D_N(N) \times Y_N$$

$$Z = D_L(L) \times Z_L + D_M(M) \times Z_M + D_N(N) \times Z_N$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

34.3.4.1 WhitePoint

The mandatory key/value pair <*WhitePoint: VectorReference*> references a 3-element Vector whose elements, of type *Number* specify the *CIE 1931 (XYZ)*-space tristimulus value of the diffuse white point. Each element shall have a value greater than 0, and the Y component shall be equal to 1. The order of elements in the Vector is:

$[\ X_w\ \ 1\ \ Z_w\ ]$

34.3.4.2 BlackPoint

The optional key/value pair <*BlackPoint: VectorReference*> references a 3-element Vector whose elements, of type *Number* specify the *CIE 1931 (XYZ)*-space tristimulus value of the diffuse black point. Each element shall have a value greater than 0, and the order of elements in the Vector is:

$[\ X_b\ \ Y_b\ \ Z_b\ ]$

If the BlackPoint key is not present, the default value used is:

[0 0 0]

34.3.4.3 RangeA

The optional key/value pair <*RangeA: VectorReference*> references a 2-element Vector whose elements, of type *Number*, specify the valid range of values for the *A* component of the color space. The elements of *RangeA* are interpreted as a pair of bounds:

$[\ A_0\ \ A_1\ ]$

The valid range of values for *A* is then defined as $A_0 \sigma A \sigma A_1$. If the RangeA key is not present, the default valid range is:

[0 1]

34.3.4.4 DecodeA

The optional key/value pair <*DecodeA: Procedure*> is used to map *A* values into values which are linear with respect to the intermediate *LMN* representation. The Procedure is called with an *A* value as an operand and shall return the corresponding linear value.

NOTE 199 Since this Procedure is called at implementation-dependent times, it shall operate as a pure function — it should not depend on any non-default context and should not affect the current context.

If the *DecodeA* key is not present, the default value used is a null Procedure which does not affect the contents of the operand stack.

34.3.4.5 MatrixA

The optional key/value pair <*MatrixA: VectorReference*> references a 3-element Vector whose elements, of type *Number*, specify the linear interpretation of the decoded *A* component of the color space with respect to the intermediate *LMN* representation. The order of the elements in the Vector is:

$[\ L_A\ \ M_A\ \ N_A\ ]$

If the MatrixA key is not present, the default matrix used is:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642776 433

[1 1 1]

34.3.4.6 RangeLMN

The optional key/value pair <*RangeLMN: VectorReference*> references a 6-element Vector whose elements, of type *Number*, specify the valid ranges of values for the *L*, *M*, and *N* components of the intermediate representation. The elements of *RangeLMN* are interpreted as three pairs of bounds, one pair for each of the three color components:

[L_0 L_1 M_0 M_1 N_0 N_1]

The valid ranges of values for each of the three color components are then defined as

$L_0 \sigma L \sigma L_1$, $M_0 \sigma M \sigma M_1$, and $N_0 \sigma N \sigma N_1$.

If the RangeLMN key is not present, the default valid ranges are:

[0 1 0 1 0 1]

34.3.4.7 DecodeLMN

The optional key/value pair <*DecodeLMN. VectorReference*> references a 3-element Vector whose elements, of type *Procedure*, are Procedures that map *LMN* values into values which are linear with respect to *CIE 1931 (XYZ)*- space. The order of the elements in the Vector is:

[D_L D_M D_N]

Each Procedure, D_L, D_M, D_N, is called with the value of *L*, *M*, or *N*, respectively, as its operand and shall return the corresponding linear value.

NOTE 200 Since these Procedures are called at implementation-dependent times, they shall operate as pure functions — they should not depend on any non-default context and should not affect the current context.

If the *DecodeLMN* key is not present, the default value used is a Vector of null Procedures which do not affect the contents of the operand stack:

[{} {} {}]

34.3.4.8 MatrixLMN

The optional key/value pair <*MatrixLMN: VectorReference*> references a 9-element Vector whose elements, of type *Number* specify the linear interpretation of the decoded *L*, *M*, and *N* components of the intermediate LMN representation with respect to the *CIE 1931 (XYZ)*-space. The order of the elements in the Vector is:

[X_L Y_L Z_I X_M Y_M Z_M X_N Y_N Z_N]

If the MatrixLMN key is not present, the default matrix used is:

[1 0 0 0 1 0 0 0 1]

34.4 Device Color Spaces

Device color spaces enable a document to specify color values that are directly related to their representation on an output device. Color values map directly (or via simple conversions) to the application of device colorants, i.e , quantities of physical dyes or intensities of phosphors. Due to the lack of calibration and therefore

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642777 37T

inherently implementation-specific nature of these color spaces, results may vary from implementation to implementation. The Device color spaces are:

— *DeviceRGB*

— *DeviceCMYK*

— *DeviceKX*

— *DeviceGray*

34.4.1 DeviceRGB

The *DeviceRGB* color space is an additive color model whose primaries are red, green, and blue. An *additive color model* is one in which the color components are primary colors which specify the spectral range in which light is emitted and the value of each component controls the strength of that emission, with 0.0 being no emission and 1.0 being a full emission.

The Object Name of the Public Object Identifier for this color space family is *color-space-family::DeviceRGB*. The Color Space Object for this color space has no parameters:

The color components required by the **SetColor** operator to specify a color in the *DeviceRGB* color space are

<B: Number>

<G: Number>

<R: Number>

where the value of each component shall be between 0.0 and 1.0, where 0.0 represents no contribution from the given primary and 1.0 represents the maximum intensity of that primary.

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for this color space is

<0.0: Real>

<0.0: Real>

<0.0: Real>

34.4.2 DeviceCMYK

The *DeviceCMYK* color space is a subtractive color model whose primaries are cyan, magenta, yellow, and black. A *subtractive color model* is one in which the color components are primary colors which specify the spectral range in which light is absorbed and the value of each component controls the strength of the absorbtion, with 0.0 being no absorbtion and 1.0 being maximal absorbtion.

The Object Name of the Public Object Identifier for this color space family is *color-space-family::DeviceCMYK*. The Color Space Object for this color space has no parameters:

The color components required by the **SetColor** operator to specify a color in the *DeviceCMYK* color space are

<K: Number>

<Y: Number>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642778 206

<*M: Number*>

<*C: Number*>

where the value of each component shall be between 0.0 and 1.0, where 0.0 represents no light absorption by the given primary and 1.0 represents the maximum absorption by that primary.

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for this color space is

<*1.0: Real*>

<*0.0: Real*>

<*0.0: Real*>

<*0.0: Real*>

34.4.3 DeviceKX

The *DeviceKX* color space is a subtractive color model whose primaries are black and an unspecified highlight color The Object Name of the Public Object Identifier for this color space family is *color-space-family::DeviceKX*. The Color Space Object for this color space has no parameters:

The color components required by the **SetColor** operator to specify a color in the *DeviceKX* color space are

<*K: Number*>

<*X: Number*>

where the value of each component shall be between 0.0 and 1.0, where 0.0 represents no light absorption by the given primary and 1.0 represents the maximum absorption by that primary.

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for this color space is

<*1.0: Real*>

<*0.0: Real*>

34.4.4 DeviceGray

The *DeviceGray* color space is an additive, one-dimensional color space in which gray values are specified. The Object Name of the Public Object Identifier for this color space family is *color-space-family::DeviceGray*. The Color Space Object for this color space has no parameters:

The color component required by the **SetColor** operator to specify a color (level of gray) in the *DeviceGray* color space is

<*Gray: Number*>

where the valid range of values of *Gray* is 0.0 to 1.0, with 0.0 used to represent black and 1.0 used to represent white.

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for this color space is

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

<0.0: Real>

34.5 Special-Purpose Color Spaces

Certain special features are also modelled as color spaces. These include color spaces for indexed color mapping and for named colors. The special-purpose color spaces are:

— *Indexed* - Lookup-table to standardized color space mapping

— *NamedColor* - Named colors

NOTE 201 There is one other special case for color spaces. The **SetPatternColor** operator installs a pattern dictionary as the current color and uses the value of the *CurrentColorSpace* Graphics State Variable as the underlying color space for the pattern.

34.5.1 Indexed

Many input and editing devices use private device-specific color lookup tables for reasons of efficiency; it is desirable to have a general method by which to specify the translation of these lookup, or index, values into colors in one of the standard color spaces. The Indexed color space provides a method for the mapping from small integers to arbitrary colors in any one of the standard color spaces, called the *base color space.*

The Object Name of the Public Object Identifier for this color space family is *color-space-family::Indexed*. The Color Space Object for this color space has three parameters:

<Lookup: Procedure or OctetString>

<HighValue: Cardinal>

<BaseColorSpaceObject: VectorReference>

The *BaseColorSpaceObject* parameter specifies the standard color space to which indices are to be mapped. It shall not be a special-purpose color space.

The parameter *HighValue* is the upper bound of the index space. Thus, the valid range of values is 0 to *HighValue.*

The final parameter, *Lookup*, is a color lookup which provides the mapping between the values of the indices and the colors of the base color space. If *Lookup* is of type *OctetString*, the length of the OctetString shall be $m \times (HighValue + 1)$, where m is the number of color components in the base color space. The semantics of *Lookup* as both a value of type *Procedure* and as a value of type *OctetString* are described below.

The color component required by the **SetColor** operator to specify a color in the Indexed color space is

<Index: Number>

If *Index* is of type *Real*, it is truncated to an Integer. If it is outside the range 0 to *HighValue*, the nearest value in the valid range is substituted for *Index*.

Then, if *Lookup* was of type *Procedure*, the value of *Index* is pushed onto the operand stack and *Lookup* is called. *Lookup* shall translate the provided *Index* into a set of color components and shall return them on the operand stack in a form and order acceptable to the **SetColor** operator in the base color space. The **SetColor** operator then operates on the returned values as though in the base color space. *Lookup* should operate as a pure function with no side-effects, and it must be able to return color component values for any *Index* in the domain 0 to *HighValue*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642780 964

Otherwise, if *Lookup* was of type *OctetString*, *Index* is multiplied by the number of color components in the base color space (hereafter referred to as *NumComp* for convenience) and the resultant value is used as an index into *Lookup*. The *NumComp* octets in *Lookup* starting at the position indicated by the multiplied *Index* are interpreted as coded values of type *Cardinal* for the *NumComp* color components of the base color space — the values are divided by 255 to yield component values in the range 0 to 1 inclusive. The **SetColor** operator then operates on the returned values as though in the base color space.

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for the Indexed color space is

<0: Integer>

34.5.2 NamedColor

It is sometimes useful to specify color by reference to an industry standard or private color specification. The *NamedColor* color space provides a generic mechanism for the specification of color in such a manner. The Object Name of the Public Object Identifier for this color space family is *color-space-family::NamedColor*. The Color Space Object for this color space has three parameters:

<TintToColor: Procedure>

<SelectColorSpace. Procedure>

<NamedColor: Identifier>

The *NamedColor* parameter is the name of the private colorant, spot color, color separation, etc. If the presentation process is capable of using this *NamedColor* directly, the two Procedure parameters are ignored

If the presentation process is incapable of using the *NamedColor* directly, the *SelectColorSpace* parameter is executed to provide the specification of an arbitrary alternate color space to which tints of the *NamedColor* are to be mapped. This Procedure shall return on the operand stack the parameters and Color Space Family Identifier operand for the alternate color space, in a form and in an order acceptable to the **SetColorSpace** operator. The tints of the *NamedColor* requested via subsequent executions of the **SetColor** operator will be interpreted relative to this alternate color space via the *TintToColor* Procedure parameter, as described below.

The color component required by the **SetColor** operator to specify a color in the NamedColor color space is

<Tint· Number>

where the valid range of *Tint* is 0.0 to 1.0 A value of 0.0 represents the application of the minimum amount of colorant, a value of 1.0 represents the application of the maximum amount of colorant, and intermediate values represent the application of intermediate amounts of colorant.

If the presentation process is incapable of using the *NamedColor* from the **SetColorSpace** operator directly, the value of *Tint* is pushed onto the operand stack and the Procedure *TintToColor* is called. *TintToColor* shall translate the provided *Tint* into a set of color components in the alternative color space provided by the *SelectColorSpace* parameter to the **SetColorSpace** operator. *TintToColor* shall return these color components on the operand stack in a form and order acceptable to the **SetColor** operator in the alternative color space The **SetColor** operator then operates on the returned values as though in the alternative color space. *TintToColor* should operate as a pure function with no side-effects, and it shall be able to return color component values for any valid value of *Tint*

The value of the *CurrentColor* Graphics State Variable (as it would be returned by the **GetColor** operator) immediately after execution of the **SetColorSpace** operator for the NamedColor color space is

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642781 8T0

<1.0: Real>

34.6 Operators

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and exception handling are defined in clause 36. In addition to the operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator. These generic exceptions and their semantics are described in 36.6.2.

34.6.1 SetColorSpace

The **SetColorSpace** operator accepts the following operands:

<ColorSpaceObject: VectorReference>

where *ColorSpaceObject* is obtained via execution of the **FindResource** operator. The order of the elements in *ColorSpaceObject* is such that, if the **VectorLoad** operator were executed with *ColorSpaceObject* as its operand, the following results would be returned on the operand stack:

<param$_n$: Any>

. . .

<param$_1$: Any>

<ColorSpaceFamilyIdentifier: OctetStringReference>

where *ColorSpaceFamilyIdentifier* is a reference to the OctetString which identifies the color space family being selected, and the remaining parameters are in the order specified by the appropriate subclause in 34.3 through 34.5 .

The **SetColorSpace** operator returns no results. The effects of executing the **SetColorSpace** operator are:

— Make the color space defined by *ColorSpaceObject* the current color space

— Set the value of the *CurrentColorSpace* Graphics State Variable to *ColorSpaceObject*

— Set the value of the *CurrentColor* Graphics State Variable to the appropriate value (see the appropriate subclause of 34.3 through 34.5)

If the Color Space Family Identifier is not known to the Content Processor, **RaiseError** shall be invoked with *UndefinedKey* as its operand. If parameter values are outside the valid range for the selected color space family, **RaiseError** shall be invoked with *RangeCheck* as its operand. If *ColorSpace Object* is not a reference to a Vector obtained via execution of the **FindResource** operator, **RaiseError** shall be invoked with *UndefinedResource* as its operand.

34.6.2 GetColorSpace

The **GetColorSpace** operator takes no operands and returns the value of the *CurrentColorSpace* Graphics State Variable.

If the **SetColorSpace** operator has not yet been executed in the current context of execution, **GetColorSpace** returns the initial value of the *CurrentColorSpace* Graphics State Variable as specified in clause 19.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642782 737

34.6.3 SetColor

The **SetColor** operator accepts the following operands:

<*comp$_n$: Any*>

. .

<*comp$_1$: Any*>

where the number of and semantics of the color component operands are dependent upon the value of the *CurrentColorSpace* Graphics State Variable, and are described in the appropriate subclause of 34.3 through 34.5. It returns no results. The effect of executing the **SetColor** operator is to set the value of the *CurrentColor* Graphics State Variable.

If the wrong number and/or type of components are provided as operands, **RaiseError** shall be invoked with *StackUnderflow* or *TypeCheck* as its operand as appropriate to the error. If any component value is outside the valid range as defined in 34.3 through 34.5, then the nearest value in the valid range will be substituted without any indication (Warning or Error).

34.6.4 GetColor

The **GetColor** operator takes no operands and returns the value of the *CurrentColor* Graphics State Variable in the form of a Mark value on the operand stack, followed by the operands to the last execution of the **SetColor** or **SetPatternColor** operator in the current context of execution, as they were presented to the **SetColor** or **SetPatternColor** operator on the operand stack

If neither the **SetColor** or **SetPatternColor** operator has been executed in the current context of execution or if **SetColorSpace** has been executed since the last execution of the above operators, **GetColor** returns the value of the *CurrentColor* Graphics State Variable relative to the current color space (see 34.3 through 34.5).

34.6.5 SetOverPrint

The **SetOverPrint** operator accepts a single operand

<*OverPrint: Boolean*>

and returns no results The effect of executing the **SetOverPrint** operator is to set the value of the *CurrentOverPrint* Graphics State Variable to the value of the *OverPrint* operand.

Some SPDL implementations may be capable of producing *separations*. Separations are renderings of the page image in which there is a separate instance of the page image for each separation colorant. Each of the separations has only the portions of the page image that affect its given colorant. The seperation colorants may be primaries, such as Cyan, Magenta, Yellow and Black, or they may be other colorants producable by the presentation device. The latter class of colorants are called *spot colors*. With NamedColor color spaces, choosing the name of a separation colorant may restrict imaging into that particular separation. Whether or not separations are produced is outside the scope of this International Standard.

NOTE 202 The most common use for separations is to produce plates for a color printing press. The most common printing presses use four color separations, one for each of the inks: Cyan, Magenta, Yellow and Black. Such presses may be able to do spot colors with any ink, however. Alternatively, a spot color may be used as a highlight color on a color printer capable of rendering two colors.

Since each separation is rendered separately, there is an option which is not available when the page image is rendered in a single step. When only some of the separations are selected for imaging, then there is an

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642783 673

option as to what happens in the other separations, the separations into which imaging is not taking place. Following the normal rules of this International Standard, inks are opaque so the corresponding areas of the other separations should be erased when imaging occurs in the selected separations. This is what happens when the *CurrentOverPrint* Graphics State Variable is false. This is the default for this variable. When the *CurrentOverPrint* Graphics State Variable is true, however, the corresponding areas of the other separations are not erased. They are not changed in any way. This allows overprinting to occur. If separations are not being produced, then the *CurrentOverPrint* Graphics State Variable is ignored.

NOTE 203 Overprinting can be used to image traps (spreads and chokes) used to compensate for small misregistrations between pass of the printing press. Traps typically underlay dark colors with light colors to insure that unwanted white does not show through when the separations are misregistered relative to one another.

34.6.6 GetOverPrint

The **GetOverPrint** operator takes no operands and returns a single result

<OverPrint: Boolean>

where *OverPrint* is the value of the *CurrentOverPrint* Graphics State Variable.

35 Rendering and Rendering Control Operators

This clause describes the model for rendering and the facilities for controlling how graphical elements and colors are rendered on an instance of the medium by the Presentation Process. The *presentation device* is the physical agent by which the output of the Presentation Process is rendered onto the media

NOTE 204 Use of most of the operators for controlling rendering requires knowing the characteristics of the presentation device. A document that is intended to be device independent should not use any of the operators described in this clause.

Rendering, in the model of the Presentation Process used in this International Standard, is carried out by the Marking and Finishing Process (see 4.3.5 and 16.6). The information that controls rendering is captured during content processing, however. This information is taken from a subset of the Graphics State Variables that contain values that control rendering: *CurrentBlackGeneration*, *CurrentColorRendering*, *CurrentHalftone*, *CurrentOverPrint*, and *CurrentUnderColorRemoval*. As each image element is placed on the page image (see 17.2), the values of the above Graphics State Variables are saved and associated with the image element to be used when the page image is rendered onto the an instance of the the media. At that time the rendering control information is used as specified below.

35.1 Model for Rendering

Nearly all of the rendering facilities that are under program control have to do with the reproduction of shades of colors. The model for rendering is logically a multiple-step process whose semantics are outlined below. Depending on the current color space and on the characteristics of the presentation device, in particular whether it supports continuous-tone output and the set of device colorants, it is not always necessary to perform every step.

— If a color has been specified in a CIE-related color space, as described in 34.3, the Presentation Process must first transform it into one of the device color spaces (*DeviceRGB*, *DeviceCMYK*, *DeviceKX*, or *DeviceGray*), whichever is appropriate for the presentation device. This transformation is controlled by a *color rendering Dictionary*.

— If a color has been specified in a device color space, as described in 34.4, that is inappropriate for a particular presentation device (for example, a color specified in the *DeviceRGB* color space and a gray-scale presentation device), the Presentation Process invokes a *color conversion function*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642784 50T

— The Presentation Process then maps the device color values through *transfer functions*, one for each device colorant (This step is sometimes called gamma correction.) The transfer functions compensate for non-linear response in the presentation device and/or in the human eye.

— If the presentation device cannot reproduce continuous tones but only certain discrete colors (such as black and white pixels), the Presentation Process invokes a *halftone function*, which approximates the desired colors by means of patterns of pixels.

— Finally, the Presentation Process performs *scan conversion* to paint the appropriate pixels on the presentation medium with the requested colors.

The remainder of this clause discusses the use of color rendering Dictionaries in converting CIE-related colors to device colors; the algorithms for conversion between different device color spaces; and the specification of transfer functions, halftone screens, and halftone functions. The operators which provide programmatic control of these various functions are described in 35.9 .

NOTE 205 This clause is intended to describe the model for rendering; it is not intended to specify underlying implementation methods. Conforming implementations must, however, provide operators which adhere to the semantics of the operators described in 35 9.

35.2 Conversions from CIE Color> to Device Color

Conversion from a CIE color value to a device color value requires two main steps:

— Adjust the CIE color value according to a *gamut mapping*. A *gamut* is a subset of all possible colors in a color space A document has a *source gamut* which consists of all colors that it uses. A presentation device has a *device gamut* which consists of all colors that it is capable of reproducing. If a document specifies colors that the device cannot reproduce, this step transforms colors from the source gamut to the device gamut.

— Generate a corresponding device color value according to a color rendering function. For a given CIE color value, this function computes a color value in the device's native color space.

The source gamut is specified by a document when it selects a CIE color space by executing **SetColorSpace**. This specification is device-independent.

The device gamut and the color rendering function are described by a *color rendering Dictionary*, which is a parameter of the Graphics State. The default color rendering Dictionary and its contents are completely implementation-dependent. The **SetColorRendering** operator installs a new color rendering Dictionary in the Graphics State; the **GetColorRendering** operator returns the current color rendering Dictionary. The gamut mapping and color rendering function are applied only to color values presented in a CIE color space. By definition, color values in device color spaces control the device color components directly.

NOTE 206 35.2.1 contains many references to the contents of color rendering Dictionaries. See 35.3 for a description of color rendering Dictionaries

35.2.1 Transforming CIE Color to Device Color

The key/value pairs in a color rendering Dictionary together define a color rendering function that transforms CIE color values to device color values. The output from the color rendering function is subject to further transformations — device color space conversion, transfer function, and halftone function.

Every color rendering Dictionary must have a *ColorRenderingType* key/value pair whose value is a Cardinal. The value specifies the architecture of the color rendering function as a whole. The remaining entries in the Dictionary are interpreted according to this value. The only color rendering Dictionaries whose semantics are

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642785 446

defined by this International Standard are those whose *ColorRenderingType* is 1. The default color rendering Dictionary on an implementation may have a *ColorRenderingType* other than 1.

The type of color rendering function represented by a *ColorRenderingType* 1 color rendering Dictionary is based on a two-stage, non-linear transformation of the *CIE 1931 (XYZ)*-space. The target space of this transformation is called the *render color space*. Values in this space can be treated in one of two ways:

— used directly as color values in the *DeviceRGB* or *DeviceGray* color space

— used to index a three-dimensional lookup table that in turn contains color values to be interpreted in the *DeviceRGB* or *DeviceCMYK* color space.

The first method is most appropriate for use with additive, linear color devices, while the second method is required for high fidelity reproduction on devices whose color rendition cannot be described by a simple formula.

Conceptually, conversion of a color value from a CIE color space to a device color space involves the following steps:

a) Transform the CIE-based color value from its original color space to *CIE 1931 (XYZ)*-space. This transformation depends on the various parameters of the color space.

b) Adjust the *X*, *Y*, and *Z* values to account for differences in the *WhitePoint* and *BlackPoint* of the source and the device. This transformation attempts to preserve color appearance and visual contrast, according to the values of the *MatrixPQR* and *TransformPQR* entries of the color rendering Dictionary. The diffuse white and black points of the source are given as the *WhitePoint* and *BlackPoint* parameters of the color space; the diffuse white and black points of the device are given as the *WhitePoint* and *BlackPoint* entries of the color rendering Dictionary. If the corresponding entries are equal, this step reduces to the identity transformation.

c) Transform the color value from *CIE 1931 (XYZ)*-space into the render color space according to the *MatrixLMN*, *EncodeLMN*, *MatrixABC*, and *EncodeABC* entries of the color rendering Dictionary, producing three components *A*, *B*, and *C*.

NOTE 207 These three components are arbitrarily named and should not be confused with the *A*, *B*, and *C* components of the *CIEBasedABC* color space.

d) If a *RenderTable* key is in the color rendering Dictionary, use the *A*, *B*, and *C* components to index into this three-dimensional lookup table, yielding an interpolated value. This value consists of 3 or 4 color components, depending on how the table has been defined. Each of these components is transformed by a procedure into a color component in device color space. If there are 3 components, they specify red, green, and blue values according to the *DeviceRGB* color space; if there are 4 components, they specify cyan, magenta, yellow, and black values according to the *DeviceCMYK* color space.

If there is no *RenderTable*, use the *A*, *B*, and *C* components as the device color value directly. If the device's native color space is *DeviceGray*, the *A* component specifies the gray value and the *B* and *C* components are ignored. Otherwise, the *A*, *B*, and *C* components specify red, green, and blue values according to the *DeviceRGB* color space.

35.3 Color Rendering Dictionaries

This subclause specifies the entries in a color rendering Dictionary. A high-level description of the use of these entries in the CIE color to device color conversion process is given in 35.2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642786 382

The mandatory key/value pair *<ColorRenderingType: Cardinal>* selects the architecture of the color rendering function as a whole. This key/value pair shall be present in all color rendering Dictionaries. The only standard *ColorRenderingType* provided by this International Standard is type 1. The mandatory and optional entries in a *ColorRenderingType* 1 color rendering Dictionary (in addition to the mandatory key/value pair *ColorRenderingType*) are described in 35.3.1.

35.3.1 Color Rendering Type 1 Dictionary

35.3.1.1 WhitePoint

The mandatory key/value pair *<WhitePoint. Vector>* is a 3-element Vector of type *Number* whose elements specify the *CIE 1931 (XYZ)*-space tristimulus value of the presentation device's diffuse white point. Each element must have a value greater than 0, and the Y component must be equal to 1. The order of elements in the Vector is:

$[\ X_w\ \ 1\ \ Z_w\ ]$

35.3.1.2 BlackPoint

The optional key/value pair *<BlackPoint: Vector>* is a 3-element Vector of type *Number* whose elements specify the *CIE 1931 (XYZ)*-space tristimulus value of the presentation device's diffuse black point. Each element must have a value greater than or equal to 0, and the order of elements in the Vector is:

$[\ X_b\ \ Y_b\ \ Z_b\ ]$

If the *BlackPoint* key/value pair is not present, the default value used is:

[0 0 0]

35.3.1.3 MatrixPQR

The optional key/value pair *<MatrixPQR: Vector>* is a 9-element Vector of type *Number* whose elements specify the linear interpretation of the encoded *X*, *Y*, and *Z* components of the *CIE 1931 (XYZ)*-space with respect to the intermediate PQR representation, as described in 35.3.1.4. The order of the elements in the Vector is:

$[\ P_X\ \ Q_X\ \ R_X\ \ P_Y\ Q_Y\ \ R_Y\ \ P_Z\ \ Q_Z\ \ R_Z\ ]$

If the *MatrixPQR* key/value pair is not present, the default matrix used is:

[1 0 0 0 1 0 0 0 1]

35.3.1.4 TransformPQR

The mandatory key/value pair *<TransformPQR: Vector>* is a 3-element Vector of type *Procedure* whose elements are Procedures that transform the *P*, *Q*, and *R* components of the intermediate PQR representation to accommodate for differences between the source's and presentation device's diffuse *WhitePoint* and *BlackPoint* while preserving color appearance and visual contrast. The order of the elements the Vector is:

$[\ T_P\ \ T_Q\ \ T_R\ ]$

Let X_{Ws}, Y_{Ws}, Z_{Ws} and X_{Bs}, Y_{Bs}, Z_{Bs} be the *CIE 1931 (XYZ)*-space tristimulus values of the source's diffuse white and black points, respectively. Let X_{Wd}, Y_{Wd}, Z_{Wd} and X_{Bd}, Y_{Bd}, Z_{Bd} be the *CIE 1931 (XYZ)*-space tristimulus values of the presentation device's diffuse white and black points, respectively. Then the source and presen-

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642787 219

© ISO/IEC

tation device tristimulus values X_s, Y_s, Z_s and X_d, Y_d, Z_d in the *CIE 1931 (XYZ)*-space , respectively, are related by the *MatrixPQR* and *TransformPQR* entries as follows:

$$P_s = X_s \times P_X + Y_s \times P_Y + Z_s \times P_Z$$

$$Q_s = X_s \times Q_X + Y_s \times Q_Y + Z_s \times Q_Z$$

$$R_s = X_s \times R_X + Y_s \times R_Y + Z_s \times R_Z$$

$$P_d = T_P(W_s, B_s, W_d, B_d, P_s)$$

$$Q_d = T_Q(W_s, B_s, W_d, B_d, Q_s)$$

$$R_d = T_R(W_s, B_s, W_d, B_d, R_s)$$

$$X_d = P_d \times X_P + Q_d \times X_Q + R_d \times X_R$$

$$Y_d = P_d \times Y_P + Q_d \times Y_Q + R_d \times Y_R$$

$$Z_d = P_d \times Z_P + Q_d \times Z_Q + R_d \times Z_R$$

where

$$W_s = [\, X_{Ws}\ Y_{Ws}\ Z_{Ws}\ P_{Ws}\ Q_{Ws}\ R_{Ws} \,]$$

$$B_s = [\, X_{Bs}\ Y_{Bs}\ Z_{Bs}\ P_{Bs}\ Q_{Bs}\ R_{Bs} \,]$$

$$W_d = [\, X_{Wd}\ Y_{Wd}\ Z_{Wd}\ P_{Wd}\ Q_{Wd}\ R_{Wd} \,]$$

$$B_d = [\, X_{Bd}\ Y_{Bd}\ Z_{Bd}\ P_{Bd}\ Q_{Bd}\ R_{Bd} \,]$$

$$P_{Ws} = X_{Ws} \times P_X + Y_{Ws} \times P_Y + Z_{Ws} \times P_Z$$

$$Q_{Ws} = X_{Ws} \times Q_X + Y_{Ws} \times Q_Y + Z_{Ws} \times Q_Z$$

$$R_{Ws} = X_{Ws} \times R_X + Y_{Ws} \times R_Y + Z_{Ws} \times R_Z$$

$$P_{Bs} = X_{Bs} \times P_X + Y_{Bs} \times P_Y + Z_{Bs} \times P_Z$$

$$Q_{Bs} = X_{Bs} \times Q_X + Y_{Bs} \times Q_Y + Z_{Bs} \times Q_Z$$

$$R_{Bs} = X_{Bs} \times R_X + Y_{Bs} \times R_Y + Z_{Bs} \times R_Z$$

$$P_{Wd} = X_{Wd} \times P_X + Y_{Wd} \times P_Y + Z_{Wd} \times P_Z$$

$$Q_{Wd} = X_{Wd} \times Q_X + Y_{Wd} \times Q_Y + Z_{Wd} \times Q_Z$$

$$R_{Wd} = X_{Wd} \times R_X + Y_{Wd} \times R_Y + Z_{Wd} \times R_Z$$

$$P_{Bd} = X_{Bd} \times P_X + Y_{Bd} \times P_Y + Z_{Bd} \times P_Z$$

$$Q_{Bd} = X_{Bd} \times Q_X + Y_{Bd} \times Q_Y + Z_{Bd} \times Q_Z$$

$$R_{Bd} = X_{Bd} \times R_X + Y_{Bd} \times R_Y + Z_{Bd} \times R_Z$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

$$\begin{pmatrix} X_P & Y_P & Z_P \\ X_Q & Y_Q & Z_Q \\ X_R & Y_R & Z_R \end{pmatrix} = \begin{pmatrix} P_X & Q_X & R_X \\ P_Y & Q_Y & R_Y \\ P_Z & Q_Z & R_Z \end{pmatrix} - 1$$

In other words, the X_s, Y_s, and Z_s components of the source color in *CIE 1931 (XYZ)*-space are treated as a three-element Vector and multiplied by *MatrixPQR* (a three by three matrix), yielding the P_s, Q_s, and R_s components of the source color with respect to the intermediate PQR representation. These components are individually transformed by the *TransformPQR* Procedures, producing the P_d, Q_d, and R_d components of the corresponding device color. Each of the components is transformed separately; there is no interaction between the components. Finally, the P_d, Q_d, and R_d components of the device color are treated as a three-element Vector and multiplied by the inverse of *MatrixPQR*. The results provide the X_d, Y_d, and Z_d componentsof the device color in the *CIE 1931 (XYZ)*-space.

As indicated by the above equations, each *TransformPQR* Procedure takes five operands

<P_s or Q_s or R_s: *Number*>

<B_d: *VectorReference*>

<W_d: *VectorReference*>

<B_s: *VectorReference*>

<W_s: *VectorReference*>

where the operands are as described in the above equations. Each interpretation of a *TransformPQR* Procedure shall produce one result

<P_d or Q_d or R_d: *Number*>

where the result is a monotonic function of its Number argument.

NOTE 208 Since these Procedures are called at implementation-dependent times, they must operate as pure functions — they should not depend on any non-default context and should not affect the current context.

35.3.1.5 RangePQR

The optional key/value pair <*RangePQR: Vector*> is a 6-element Vector of type *Number* whose elements specify the range of valid values for the *P*, *Q*, and *R* components. The elements of *RangePQR* are interpreted as three pairs of bounds, one pair for each of the three color components:

[P_0 P_1 Q_0 Q_1 R_0 R_1]

The ranges of valid values for each of the three color components are then defined as

$P_0 \sigma P \sigma P_1$, $Q_0 \sigma Q \sigma Q_1$, and $R_0 \sigma R \sigma R_1$.

If the *RangePQR* key is not present, the default ranges are:

[0 1 0 1 0 1]

35.3.1.6 MatrixLMN

The optional key/value pair <*MatrixLMN: Vector*> is a 9-element Vector of type *Number* whose elements specify the linear interpretation of the *X*, *Y*, and *Z* components of the *CIE 1931 (XYZ)*-space with respect to the intermediate *LMN* representation, as described in 35.3.1.7. The order of the elements in the Vector is:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642789 091

$[\ L_X\ M_X\ N_X\ L_Y\ M_Y\ N_Y\ L_Z\ M_Z\ N_Z\ ]$

If the *MatrixLMN* key is not present, the default matrix used is:

[1 0 0 0 1 0 0 0 1]

35.3.1.7 EncodeLMN

The optional key/value pair *<EncodeLMN: Vector>* is a 3-element Vector of type *Procedure* whose elements are Procedures that encode the *L*, *M*, and *N* components of the intermediate LMN representation. The order of the elements in the Vector is:

$[\ E_L\ E_M\ E_N\ ]$

Each Procedure is called with an unencoded *L*, *M*, or *N* value as an operand and must return the corresponding encoded value.

NOTE 209 Since these Procedures are called at implementation-dependent times, they must operate as pure functions — they should not depend on any non-default context and should not affect the current context.

If the *EncodeLMN* key is not present, the default value used is a reference to a Vector of null Procedures which do not affect the contents of the operand stack:

[{} {} {}]

The transformation defined by the *MatrixLMN* and *EncodeLMN* entries is:

$L = E_L(X \times L_X + Y \times L_Y + Z \times L_Z)$

$M = E_M(X \times M_X + Y \times M_Y + Z \times M_Z)$

$N = E_N(X \times N_X + Y \times N_Y + Z \times N_Z)$

35.3.1.8 RangeLMN

The optional key/value pair *<RangeLMN: Vector>* is a 6-element Vector of type *Number* whose elements specify the range of valid values for the *L*, *M*, and *N* components of the intermediate representation. The elements of *RangeLMN* are interpreted as three pairs of bounds, one pair for each of the three color components:

$[\ L_0\ L_1\ M_0\ M_1\ N_0\ N_1\ ]$

The ranges of valid values for each of the three color components are then defined as

$L_0 \sigma L \sigma L_1$, $M_0 \sigma M \sigma M_1$, and $N_0 \sigma N \sigma N_1$.

If the *RangeLMN* key is not present, the default ranges are:

[0 1 0 1 0 1]

35.3.1.9 MatrixABC

The optional key/value pair *<MatrixABC: Vector>* is a 9-element Vector of type *Number* whose elements specify the linear interpretation of the encoded *L*, *M*, and *N* components of the intermediate LMN representation with respect to the render color space, as described in 35.3.1.10. The order of the elements in the Vector is:

$[\ A_L\ B_L\ C_L\ A_M\ B_M\ C_M\ A_N\ B_N\ C_N\ ]$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642790 803

If the *MatrixABC* key is not present, the default matrix used is:

[1 0 0 0 1 0 0 0 1]

35.3.1.10 EncodeABC

The optional key/value pair <*EncodeABC: Vector*> is a 3-element Vector of type *Procedure* whose elements are Procedures that encode the *A*, *B*, and *C* components of the render color space. The order of the elements in the Vector is:

[E_A E_B E_C]

Each Procedure is called with an unencoded *A*, *B*, or *C* value as an operand and must return the corresponding encoded value.

NOTE 210 Since these Procedures are called at implementation-dependent times, they must operate as pure functions — they should not depend on any non-default context and should not affect the current context.

If the *EncodeABC* key is not present, the default value used is a reference to a Vector of null Procedures which do not affect the contents of the operand stack:

[{} {} {}]

The transformation defined by the *MatrixABC* and *EncodeABC* entries is:

$A = E_A (L \times A_L + M \times A_M + N \times A_N)$

$B = E_B (L \times B_L + M \times B_M + N \times B_N)$

$C = E_C (L \times C_L + M \times C_M + N \times C_N)$

35.3.1.11 RangeABC

The optional key/value pair <*RangeABC: Vector*> is a 6-element Vector of type *Number* whose elements specify the range of valid values for the *A*, *B*, and *C* components. The elements of *RangeABC* are interpreted as three pairs of bounds, one pair for each of the three color components:

[A_0 A_1 B_0 B_1 C_0 C_1]

The ranges of valid values for each of the three color components are then defined as

$A_0 \sigma A \sigma A_1$, $B_0 \sigma B \sigma B_1$, and $C_0 \sigma C \sigma C_1$.

If there is no *RenderTable* key, these ranges must lie within the range 0 to 1, since the render color space maps directly onto a device color space. If *RenderTable* is present, these ranges define the boundaries of the three-dimensional lookup table.

If the *RangeABC* key is not present, the default ranges are:

[0 1 0 1 0 1]

35.3.1.12 RenderTable

The optional key/value pair <*RenderTable: Vector*> is a Vector whose elements describe a three-dimensional lookup table and associated Procedures which together map colors in render color space to colors in device color space via table lookup and interpolation. The Vector takes the form:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642791 74T

[<N_A : Cardinal> <N_B: Cardinal> <N_C: Cardinal> <table: Vector> <m: Cardinal>
<T_1: Procedure> <T_2: Procedure> ... <T_m: Procedure>]

where the dimensions of the lookup table are given by N_A, N_B, and N_C; each of which must have an Integer greater than 1. Each entry in the table consists of m encoded color component values, where the value of m is 3 or 4. The entry at integer coordinates (a, b, c) in the table, where $0 \sigma a < N_A$, $0 \sigma b < N_B$, and $0 \sigma c < N_C$, contains the encoded device color value that corresponds to render color space components A, B, and C, where:

$$A = A_0 + a \times (A_1 - A_0) / (N_A - 1)$$

$$B = B_0 + b \times (B_1 - B_0) / (N_B - 1)$$

$$C = C_0 + c \times (C_1 - C_0) / (N_C - 1)$$

and the values A_0, A_1, B_0, B_1, C_0, and C_1 are those given in the *RangeABC* entry.

The element *table* must be a Vector of N_A OctetStrings which define the contents of the lookup table. Each OctetString must contain $m \times N_B \times N_C$ octets. Within the String at index a in the Vector, the m octets starting at position $m \times (b \times N_C + c)$ comprise the table entry at location (a, b, c). These octets are interpreted as encoded device color components e_1, e_2, ... e_m.

The elements T_1, T_2, T_m are Procedures that transform the encoded components to actual device color component values. These transformations are:

$$d_1 = T_1 (e_1/255)$$

$$d_2 = T_2 (e_2/255)$$

$$d_m = T_m (e_m/255)$$

In other words, the color rendering function treats an octet extracted from the table as an Integer, divides it by 255 (producing a number in the range 0 to 1), and pushes it on the Operand Stack. It then calls the appropriate T Procedure, which is expected to consume its operand and produce a result in the range 0 to 1.

NOTE 211 Since these Procedures are called at implementation-dependent times, they must operate as pure functions — they should not depend on any non-default context and should not affect the current context.

The values d_1, d_2, d_m constitute the final device color value. That is, if m is 3 then d_1, d_2, and d_3 are the red, green, and blue components; if m is 4 then d_1, d_2, d_3, and d_4 are the cyan, magenta, yellow, and black components.

35.4 Conversions among Device Color Spaces

The device colorants used by the presentation device define a native device color space which typically corresponds to one of the color spaces *DeviceGray*, *DeviceRGB*, or *DeviceCMYK*. If the presentation device supports continuous-tone output, reproduction occurs directly; otherwise, it is accomplished by means of a halftone function. If a document specifies colors in the native device color space, no conversions are necessary. Otherwise, the Presentation Process must convert color values as specified in a document to the appropriate color values for the native color space of the presentation device. The conversion is performed by a color conversion function. The color conversion function is actually a set of functions, one for each colorant in the native device color space.

Device color conversions ordinarily are not performed when a document specifies colors in CIE color spaces; the CIE color rendering functions (see 35.2) map directly to the native device color space. Therefore this clause describes only the conversions from one device color space to another. Conversions to the *DeviceKX* color

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642792 686

space from other device color spaces, and from *DeviceKX* to other device color spaces, are not defined by this International Standard.

The conversions described in the following subclauses do not involve the use of transfer functions or halftone functions. When colors are actually to be rendered on the media, the transfer functions and halftone functions are applied at a later stage to the output of the color conversion step.

35.4.1 Conversion between DeviceRGB and *DeviceGray*

Black, white, and intermediate shades of gray can be considered special cases of *DeviceRGB* color; the color conversion function which maps a specific *DeviceGray* color value *gray* to the equivalent *DeviceRGB* color value is:

red = *gray*

green = *gray*

blue = *gray*

The color conversion function which maps a given *DeviceRGB* color value to a *DeviceGray* color value is computed according to the NTSC video standard formula:

gray = *.3* × *red* + *.59* × *green* + *.11* × *blue*

35.4.2 Conversion between DeviceCMYK and *DeviceGray*

A *DeviceGray* color value is the complement of the black component of the corresponding *DeviceCMYK* color value. The color conversion function which maps a specific *DeviceGray* color value *gray* to the equivalent the *DeviceCMYK* color value is:

cyan = *0.0*

magenta = *0.0*

yellow = *0.0*

black = *1.0* - *gray*

The color conversion function which maps a given *DeviceCMYK* color value to a *DeviceGray* color value is essentially a conversion from *DeviceCMYK* to *DeviceRGB*, followed by a conversion from *DeviceRGB* to *DeviceGray*. These two color conversion functions may be combined and expressed as:

gray = *1 0* - *min(1 0, 3* × *cyan* + *59* × *magenta* + *.11* × *yellow* + *black)*

where *min* is a function which returns the minimum value from a set of values.

35.4.3 Conversion from DeviceRGB to DeviceCMYK

The color conversion function which maps a given *DeviceRGB* color value to a *DeviceCMYK* color value is defined by two sets of equations. The first set of equations express the relationship between red-green-blue and cyan-magenta-yellow. The second set of equations specify how *black generation* and *undercolor removal* are used to generate a black component and alter the other components to produce a better approximation of the original color.

The subtractive color primaries cyan, magenta, and yellow are the complements of the additive primaries red, green, and blue; so in theory, the color conversion function should be very simple:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642793 512

cyan = 1.0 - *red*

magenta = 1.0 - *green*

yellow = 1.0 - *blue*

Logically, cyan, magenta, and yellow are all that are needed to generate a printing color. Thus, an equal percentage of cyan, magenta, and yellow should create the equivalent percentage of black.

35.4.3.1 Black Generation and UnderColorRemoval

In reality, colored printing inks do not mix perfectly; such combinations often form dark brown shades instead. Thus, it is often desirable to substitute real black ink for the black portion of a color to obtain a truer color rendition. Computing the quantity of this component requires some additional equations.

— A *black generation* function calculates the amount of black to be used when trying to reproduce a particular color.

— An *undercolor removal* function reduces the amount of cyan, magenta, and yellow components to compensate for the amount of black that was added by the black generation function.

SPDL provides limited control over black generation and undercolor removal when converting from *DeviceRGB* to *DeviceCMYK* color spaces. Applications requiring finer control must specify colors in CIE color spaces and control conversion to *DeviceCMYK* color by means of the CIE color rendering Dictionary (see 35.2). The complete conversion from *DeviceRGB* color to *DeviceCMYK* color is as follows, where *BG (k)* and *UCR (k)* are invocations of the black generation and undercolor removal functions, respectively. The first set of equations define c, m, y and b, the initial approximation to the *cyan*, *magenta*, *yellow* and *black* values.

c = 1.0 - *red*

m = 1.0 - *green*

y = 1.0 - *blue*

k = *min(c, m, y)*

The second set of equations use the initial approximations to compute the actual *cyan*, *magenta*, *yellow* and *black* values.

cyan = *min*(1.0, *max*(0.0,*c* - *UCR(k)*))

magenta = *min*(1.0, *max*(0.0, *m* - *UCR(k)*))

yellow = *min*(1.0, *max*(0.0, *y* - *UCR(k)*))

black = *min*(1.0, *max*(0.0, *BG(k)*))

where *min* is a function which returns the minimum value from a set of values and *max* is a function which returns the maximum value from a set of values.

The black generation and undercolor removal functions are defined as Procedures and are stored in the Graphics State Variables *CurrentBlackGeneration* and *CurrentUnderColorRemoval*, respectively. The **SetBlackGeneration** and **SetUnderColorRemoval** operators set these parameters in the Graphics State.

The default values for the *CurrentBlackGeneration* and *CurrentUnderColorRemoval* Graphics State Variables are implementation-dependent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642794 459

The Presentation Process calls these Procedures when it needs to perform *DeviceRGB* color to *DeviceCMYK* color conversion. Each Procedure is called with a single numeric operand and is expected to return a single numeric result. Because the Procedures may be called at unpredictable times, they must operate as pure functions, without side-effects.

The operand of both Procedures is *k*, the minimum of the intermediate *c*, *m*, and *y* values that have been computed by subtracting the original red, green, and blue values from 1. Nominally, *k* is the amount of black that can be removed from the cyan, magenta, and yellow components and be substituted as a separate black component. The black generation function computes the actual black component as a function of the nominal *k* value.

The undercolor removal function computes the amount that is to be subtracted from each of the intermediate *c*, *m*, and *y* values to produce the final cyan, magenta, and yellow components; it is expected to return values in the range -1 to 1 (inclusive). If the undercolor removal function returns a negative value, the absolute value of the returned value is added to each component. The component values that result from black generation and undercolor removal are expected to be in the range 0 to 1. If a value falls outside this range, the nearest valid value is substituted automatically, without error indication, as indicated by the *min* and *max* functions above.

35.4.4 Conversion from DeviceCMYK to DeviceRGB

The color conversion function which maps a given *DeviceCMYK* color value to a *DeviceRGB* color value is:

$red = 1.0 - min(1.0, cyan + black)$

$green = 1.0 - min(1.0, magenta + black)$

$blue = 1.0 - min(1.0, yellow + black)$

where *min* is a function which returns the minimum value from a set of values. This is a simple set of equations that do not involve the black generation or undercolor removal functions.

35.5 Transfer Functions

A transfer function adjusts the values of color components to compensate for non-linear response in a presentation device and in the human eye. Each component of a device color space (for example, the red component of the *DeviceRGB* color space) is intended to represent the perceived lightness or intensity of that component in proportion to the numeric value. Many devices do not exhibit such a linear response; a transfer function can compensate for the actual response of the device. This mapping is sometimes called *gamma correction*.

In the sequence of steps for processing colors, the Presentation Process applies the transfer function after performing conversions between color spaces (if necessary) but before applying the halftone function (if necessary). A separate transfer function applies to each device color component; there is no interaction between components. Which device color components are relevant is indicated by the *ProcessColorClass* State Variable. Transfer functions operate in the native color space of the output device, regardless of the color space in which colors were originally specified.

35.5.1 Specifying Transfer Functions

Transfer functions are specified within a halftone Dictionary. Each transfer function is the value bound to the *TransferFuntion* key in an appropriate halftone Dictionary (see 35.7). The currently active transfer functions are in the halftone Dictionary or Dictionaries that are referenced by the value of the Graphics State Variable *CurrentHalftone*. The value of this Graphics State Variable may be changed with the **SetHalftone** operator.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642795 395

NOTE 212 If the device only has one device color, then there is only one halftone Dictionary referenced by *CurrentHalftone*. If the device has more than one device color, then there is a Type 5 halftone Dictionary with one key per device color (see 19.18.1). The halftone Dictionaries bound to those keys have the transfer functions for each device color.

The default values of the *TransferFunction* keys of the halftone Dictionary or Dictionaries referenced by the *CurrentHalftone* Graphics State Variable is implementation-dependent.

Transfer functions are represented by Procedures. Transfer function Procedures are executed with an operand of type Number in the range 0.0 to 1.0 (inclusive), and that must return a Number in the same range. The operand is the value of a color component in the native device color space (either specified directly or produced by conversion from some other color space). The result is the transformed value that is to be transmitted to the device (after halftoning, if necessary).

The operand and result of a transfer function are always interpreted as if the component were additive (red, green, blue, or gray); that is, larger numbers indicate lighter colors. If the component is subtractive (cyan, magenta, yellow, or black), the Presentation Process converts it to additive form by subtracting it from 1.0 before passing it to the transfer function. The result from the transfer function is always in additive form; it is passed on to the halftone function in this form. This convention also applies to *NamedColor* components.

NOTE 213 Since these Procedures are called at implementation-dependent times, they must operate as pure functions — they should not depend on any non-default context and should not affect the current context.

35.6 Halftone Functions

Halftoning is the process by which continuous-tone colors are approximated by a pattern of pixels that can achieve only a limited number of discrete colors. The most familiar case of this is the rendering of gray tones with black and white pixels. Some devices can reproduce continuous-tone colors directly. For such devices, halftoning is not required.

If halftoning is required, it occurs after all color components have been transformed by the appropriate transfer function. For each device pixel, the *halftone function* determines which of the colors the device is able to produce will be used for that pixel. The input to the halftone function consists of continuous-tone gamma-corrected color components in the presentation device's native color space. The output of the halftone function consists of a color that the device can actually render on an instance of the media. The halftone function is implicitly defined by reference to the model for halftone screens.

NOTE 214 As with most rendering controls, halftone functions are inherently device-specific, and should only be used in a document in a known local environment where extremely high-quality output requirements necessitate the use of such controls. Documents which do not require the use of halftone functions for extremely high-quality output should refrain from defining their own halftone functions, and rely instead upon the implementation-specific default halftone functions which are present in all conforming implementations.

35.6.1 Model for Halftone Screens

The halftone functions supported by SPDL are based on the use of a halftone screen. A screen is defined by conceptually laying a uniform square grid of *halftone cells* over the device pixel array. Each pixel belongs to one cell of the grid; a halftone cell usually contains many device pixels. The screen grid is defined entirely in device pixel space, unchanged by modifications to the *CurrentTransformation*, this property is essential for ensuring that adjacent areas colored by halftone functions are properly stitched together without *seams*.

For a black-and-white device, each cell of a screen can be made to approximate a shade of gray by painting some of the cell's pixels black and some pixels white. Numerically, the gray level produced within a cell is the ratio of the cell's pixels that are white to the total number of pixels in that cell. If a cell contains n pixels then it is capable of rendering $n + 1$ different gray levels. A particular desired gray value g in the range 0 to 1 is produced by making i pixels white, where $i = floor(g \times n)$.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642796 221

The foregoing description also applies to color output devices whose pixels consist of primary colors that are either completely on or completely off. A halftone function is applied to each color component independently, producing shades of that color.

Color component values are presented to the halftone functions in additive form, regardless of whether they were originally specified in additive (RGB or gray) or subtractive (CMYK) form. Larger values of a color component represent lighter colors — greater intensity in an additive device; less ink in a subtractive device. Transfer functions are always interpreted as if the component were additive; see 35.5.1.

35.6.2 Generating Halftone Screens

There are two basic methods of generating halftone screens available in all implementations:

- The document may specify a *screen frequency*, *screen angle*, and *spot function* for each color component; this method is the default method for generating halftone screens and is discussed in 35.6.2.1.

- The document may specify a Vector of threshold values to be tiled across device pixel space; these values are consulted to determine how to render intermediate color levels. This method is most useful for low-resolution devices and is discussed in 35.6.2.2.

35.6.2.1 Spot Functions

NOTE 215 For ease of description, the explanation of this method of generating halftone screens employs as an example a black-and-white device producing gray levels. The same processes and algorithms are employed on a color device to render intermediate colors, that is, colors which cannot be directly represented on the device

The default method for generating halftone screens requires a *screen frequency*, *screen angle*, and *spot function* for each color component. The halftone screen grid has a *frequency* (number of halftone cells per centimeter) and *angle* (orientation of the grid lines relative to the device pixel coordinate system). As a cell's desired gray value varies from black to white, individual pixels in the cell change from black to white in a well-defined sequence. If a particular gray includes certain white pixels, lighter grays will include the same white pixels as well as some additional ones. The order in which pixels change from black to white for increasing gray levels is specified by a *spot function*, which is defined as a Procedure. The spot function describes the order of pixel whitening in an indirect way that minimizes interactions with screen frequency and angle.

A halftone cell has its own coordinate system: the center of the square is the origin and the corners are at ± 1 in x and y. In this system, each pixel in the cell is centered at x and y coordinates that are both between -1 and 1 For each pixel, the spot function Procedure is called with the coordinates of the pixel as an operand; the Procedure must return a single number between -1 and 1 that defines the pixel's position in the ordering.

The actual values returned by the spot function are not significant; the order in which pixels are whitened is determined solely by the relative spot function values for different pixels. As a cell's gray value varies from black to white, the first pixel whitened is the one whose spot function has the lowest (most negative) value, the next pixel is the one with the next higher spot function value, and so on. If two pixels have the same spot function value, the assignment of their relative order is implementation-dependent.

35.6.2.2 Threshold Arrays

An alternate method of generating halftone screens is via an array of threshold values that directly control individual device pixels in a halftone cell. This method is useful on low-resolution devices because it provides a high degree of control over halftone rendering and because it permits halftone cells to be rectangular, whereas halftone cells defined by a spot function are always square.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

A *threshold array* is much like a sampled raster graphics image: it is a rectangular array of pixel values. However, it is defined entirely in device pixel space and the sample values are always octets. The pixel values nominally represent gray levels where 0 is black and 255 is white. The threshold array is replicated to tile the entire device pixel space; thus, each pixel of device space is mapped to a particular sample of the threshold array.

On a bi-level device, where each pixel is either black or white, the halftone function implicitly defined by a threshold array is created as follows: For each device pixel that is to be painted with some gray level, consult the corresponding pixel of the threshold array. If the desired gray level is less than the pixel value in the threshold array, paint the device pixel black; otherwise paint it white. Gray values in the range 0 to 1 (inclusive) correspond to pixel values 0 to 255 (inclusive) in the threshold array.

This scheme easily generalizes to monochrome presentation devices which can render multiple levels per pixel. For example, a presentation device may be to directly render one of four different gray levels: black, dark gray, light gray, and white, on each device pixel. For each device pixel that is to be painted with some intermediate gray level that is not one of the four directly representable by a device pixel, the algorithm consults the corresponding element of the threshold array to determine whether to use the next lower or next higher representable gray level.

A halftone function defined in this way can also be used with color displays that have a limited number of values for each color component. The red, green, and blue values are simply treated independently as if they were gray levels.

35.7 Halftone Dictionaries

A *halftone Dictionary* is a Dictionary object whose key/value pairs are parameters used to defined one or more halftone functions. The Graphics State includes a current halftone Dictionary, which specifies the halftone functions to be used in rendering. The operator **GetHalftone** returns the current halftone Dictionary; the operator **SetHalftone** establishes a different halftone Dictionary as the current one. Other operators may affect the current halftone Dictionary in the Graphics State; see 35.8.

Every halftone Dictionary must have a *HalftoneType* key whose value is a Cardinal. This specifies the major type of halftone function; the remaining entries in the Dictionary are interpreted according to the type. Three types of standard halftone Dictionaries are specified in this International Standard. The following subclauses describe the standard contents of each standard type of halftone Dictionary.

NOTE 216 This International Standard describes only the mandatory and optional standard entries in halftone Dictionaries. There may be other implementation-dependent entries in halftone Dictionaries which are not defined in this International Standard

35.7.1 HalftoneType 1 Halftone Dictionaries

A halftone Dictionary whose *HalftoneType* is 1 defines a halftone screen according to a frequency, angle, and spot function (see 35.8). A *HalftoneType* 1 halftone Dictionary can be made the current halftone Dictionary in the Graphics State by the **SetHalftone** operator. The contents of a *HalftoneType* 1 halftone Dictionary are described in the following subclauses.

35.7.1.1 HalftoneType

The mandatory key/value pair *<HalftoneType: Cardinal>* must have a value of 1.

35.7.1.2 Frequency

The mandatory key/value pair *<Frequency: Number>* specifies the number of halftone cells per centimeter.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642798 0T4

35.7.1.3 Angle

The mandatory key/value pair *<Angle: Number>* specifies the number of degrees by which the halftone screen is to be rotated with respect to the RCS.

35.7.1.4 SpotFunction

The mandatory key/value pair *<SpotFunction: Procedure>* defines the order in which device pixels within a halftone cell are adjusted for different gray/color levels. The Procedure takes two operands

<y. Number>

<x: Number>

and returns one result

<relval: Real>

where *x* and *y* are the coordinates of a device pixel in the halftone cell coordinate system and *relval* is a Real between -1.0 and 1.0 (inclusive) that defines the position of the pixel in the ordering. See 35.6.2.1 for a detailed description of the semantics of spot functions.

35.7.1.5 TransferFunction

The mandtory key/value pair *<TransferFunction: Procedure>* specifies a mapping of device color values to compensate for non-linear response in the presentation device and/or in the human eye The Procedure takes one operand

<d· Number>

and returns one result

<t· Real>

where *d* is a color value for a native device color and *t* is the value that will actually be transmitted to the device See 35.5 for a detailed description of transfer functions.

35.7.1.6 ActualFrequency

The optional key/value pair *<ActualFrequency: Number>* is a placeholder whose value is set by the **SetHalftone** operator. **SetHalftone** may make slight adjustments to the requested screen frequency so as to ensure that the patterns of enclosed pixels remain constant as the screen cells are replicated over the entire page. If this key/value pair is present in a *HalftoneType* 1 halftone Dictionary presented to the **SetHalftone** operator, the operator replaces its value with the actual frequency that was achieved

35.7.1.7 ActualAngle

The optional key/value pair *<ActualAngle: Number>* is a placeholder whose value is set by the **SetHalftone** operator. SetHalftone may make slight adjustments to the requested screen angle so as to ensure that the patterns of enclosed pixels remain constant as the screen cells are replicated over the entire page. If this key/value pair is present in a *HalftoneType* 1 halftone Dictionary presented to the **SetHalftone** operator, the operator replaces its value with the actual angle that was achieved.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642799 T30

35.7.2 HalftoneType 3 Halftone Dictionaries

A halftone Dictionary whose *HalftoneType* is 3 defines a halftone screen as an array of threshold values that directly control individual device pixels in a halftone cell. A *HalftoneType* 3 halftone Dictionary can be made the current halftone Dictionary in the Graphics State by the **SetHalftone** operator. The contents of a *HalftoneType* 3 halftone Dictionary are described in the following subclauses.

35.7.2.1 HalftoneType

The mandatory key/value pair *<HalftoneType: Cardinal>* must have a value of 3.

35.7.2.2 Width

The mandatory key/value pair *<Width: Cardinal>* specifies the width of the threshold array in pixels.

35.7.2.3 Height

The mandatory key/value pair *<Height: Cardinal>* specifies the height of the threshold array in pixels.

35.7.2.4 Thresholds

The mandatory key/value pair *<Thresholds: OctetStringReference>* specifies the threshold array. The OctetString must contain Width × Height elements describing the rectangular threshold array starting at the lower left corner (viewing the rectangular array as it will be tiled onto device pixel space) and continuing in a left-to-right, bottom-to-top order. See 35.6.2.2 for a detailed description of threshold arrays and the semantics of the values in the threshold array.

35.7.2.5 TransferFunction

The mandtory key/value pair *<TransferFunction: Procedure>* specifies a mapping of device color values to compensate for non-linear response in the presentation device and/or in the human eye. The Procedure takes one operand

<d: Number>

and returns one result

<t: Real>

where *d* is a color value for a native device color and *t* is the value that will acually be transmitted to the device. See 35.5 for a detailed description of transfer functions.

35.7.3 HalftoneType 5 Halftone Dictionaries

A halftone Dictionary whose *HalftoneType* is 5 allows specification of individual halftone screens for an arbitrary number of color components. It contains entries whose names are device color components or *NamedColor* names and whose values are halftone Dictionaries.

Which set of Names is meaningful as the set of keys for a halftone Dictonary whose *HalftoneType* is 5 depends on the value of the *ProcessColorClass* key of the value of the *DeviceDescription* State Variable whose definition in 19.18 also specifies which device color names, also called process colorants, are relevant for each type of device color space. In the case of the *NamedColor* color spaces the names correspond to the names used as *NamedColor* parameters to the **SetColorSpace** operator. Each key which is the name of a process or *NamedColor* colorant is bound to a type 1 or type 3 halftone Dictionaries which describes the halftone and transfer function for that colorant. This is in contrast to a type 1 or typehalf 3 tone Dictionary appearing as the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642800 582

main halftone Dictionary (operand to the **SetHalftone** operator), in which case that halftone Dictionary applies to all color components.

A *HalftoneType* halftone Dictionary can be made the current halftone Dictionary in the Graphics State by the **SetHalftone** operator. In addition to the arbitrary number of optional color name/halftone Dictionary pairs, a *HalftoneType* 5 halftone Dictionary must contain two mandatory entries; these are described in the following subclauses.

35.7.3.1 HalftoneType

The mandatory key/value pair *<HalftoneType: Cardinal>* must have a value of 5.

35.7.3.2 Default

The mandatory key/value pair *<Default: Dictionary>* is a halftone Dictionary specifying the halftone function to be applied to any colorant that does not have its own key in the *HalftoneType* 5 halftone Dictionary.

35.8 Defining Halftone Functions

Halftone functions are implicitly defined by reference to the model for halftone screens (see 35.6.1). The information used in defining the current halftone function or functions is stored in the Graphics State Variable *CurrentHalftone*. The value of this Graphics State Variable can be set by the **SetHalftone** operator. The initial value of this Graphics State Variable is specified in clause 19

35.9 Operators

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. Content exceptions and exception handling are defined in clause 36 In addition to the operator-specific exceptions, there are generic exceptions which may be raised during the interpretation of almost any operator These generic exceptions and their semantics are described in 36.6.2.

35.9.1 CIE Color Conversion Operators

The following operators provide control over the conversion from CIE-related color space colors to device color space colors

35.9.1.1 SetColorRendering

The **SetColorRendering** operator takes one operand

<crd: DictionaryReference>

and returns no results. *crd* must be a reference to a color rendering Dictionary (as described in 35.3). The effect of this operator is to set the value of the *CurrentColorRendering* Graphics State Variable to *crd*.

35.9.1.2 GetColorRendering

The **GetColorRendering** operator takes no operands and returns one result

<crd: DictionaryReference>

where *crd* is the value of the *CurrentColorRendering* Graphics State Variable; the current color rendering Dictionary in (see 35.3) the Graphics State.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642801 419

35.9.2 Device Color Conversion Operators

The following operators provide control over the conversion from RGB device colors to CMYK device colors.

35.9.2.1 SetUnderColorRemoval

The **SetUnderColorRemoval** operator takes one operand

<ucr: Procedure>

and returns no results. *ucr* must be an undercolor removal Procedure (as described in 35.4.3.1). The effect of this operator is to set the value of the *CurrentUnderColorRemoval* Graphics State Variable to *ucr*.

35.9.2.2 GetUnderColorRemoval

The **GetUnderColorRemoval** operator takes no operands and returns one result

<ucr: DictionaryReference>

where *ucr* is the value of the *CurrentUnderColorRemoval* Graphics State Variable.

35.9.2.3 SetBlackGeneration

The **SetBlackGeneration** operator takes one operand

<bg: Procedure>

and returns no results. *bg* must be a black generation Procedure (as described in). The effect of this operator is to set the value of the *CurrentBlackGeneration* Graphics State Variable to *bg*.

35.9.2.4 GetBlackGeneration

The **GetBlackGeneration** operator takes no operands and returns one result

<bg: DictionaryReference>

where *bg* is the value of the *CurrentBlackGeneration* Graphics State Variable.

35.9.3 Halftone Function Operators

The following operators provide control over the halftone function of the model for rendering.

35.9.3.1 SetHalftone

The **SetHalftone** operator takes one operand

*<htd: Dictionary*Reference>

and returns no results. The effect of executing the **SetHalftone** operator is to set the value of the *CurrentHalftone* Graphics State Variable to the value of the halftone DictionaryReference operand *htd*, and to make the referenced Dictionary readonly.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642802 355

35.9.3.2 GetHalftone

The **GetHalftone** operator takes no operands and returns one result

<htd: DictionaryReference>

where *htd* is a reference to a halftone Dictionary that is the value of the *CurrentHalftone* Graphics State Variable in the Graphics State.

36 Exception Handling

During structure processing or content processing, various exception conditions may arise. This clause defines the set of exceptions that may arise, and the exception handling mechanisms of the SPDL presentation process. Exception handling provides a way to continue processing at some relevant point either avoiding the action that caused the exception or mitigating its effects.

Exceptions are defined and handled at two levels. Structure exceptions arise and are handled at the BLOCK level. The exceptions that may arise in structure are defined in 36.1. Structure exception handling is defined in 36.2. Content exceptions arise at the content level. The exceptions that may arise are defined in 36.3. Content exception handling is defined in 36.4 . Operators having to do with content exceptions and content exception handling are defined in 36.5. Interpreter errors form a special class of exception conditions that may arise in content These are defined in 36 6.

36.1 Structure Exceptions

A structure exception arises within a BLOCK, upon any of the following occurrances:

— syntax error in a structure subordinate to the BLOCK;

— illegal specification in a structure subordinate to the BLOCK;

— DPI constraint violation in a structure subordinate to the BLOCK;

— structure warning in a structure subordinate to the BLOCK;

— implementation constraint violation in a structure subordinate to the BLOCK;

— unhandled content exception arising in content subordinate to the BLOCK;

— unhandled structure exception arising in a BLOCK subordinate to the BLOCK.

When any of the above violations occur, an exception is raised in the BLOCK most immediately superior to the structure in which the violation occurs.

36.1.1 Syntax error in structure

A syntax error arises in a BLOCK when the structure processor encounters structures or parts thereof, subordinate to the BLOCK, which violate the syntax prescribed in clause 37 and clause 38 of this International Standard.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

36.1.2 Illegal specification in structure

An illegal specification exception arises in a BLOCK when the structure processor encounters structure elements, subordinate to the BLOCK, which specify a value or values which are not allowed by this International Standard.

36.1.3 Implementation constraint violation

An implementation constraint violation arises in a BLOCK when the structure processor encounters structure elements, subordinate to the BLOCK, which specify values which are not supported by the implementation.

NOTE 217 In a conforming implementation, this exception can only arise in regard to specific limits as defined in this International Standard; see clause 39.

36.1.4 Unhandled content exception in content

An unhandled content exception arises in a BLOCK when the Content Processor encounters an unhandled content exception in content subordinate to the BLOCK. (See 36.3 and 36.4 for content exceptions and content exception handling.)

36.1.5 Unhandled structure exception in subordinate BLOCK

When any of the above five structure exceptions occur in a BLOCK, if that BLOCK does not handle the exception (see 36.2), then the same structure exception arises in the most immediately superior BLOCK.

If an unhandled structure exception occurs in the DOCUMENT (to which there is no superior BLOCK) then the exception is handled as specified in 36.2.2 .

36.2 Structure Exception Handling

The Block State of each BLOCK has an *abort-policy* presentation parameter, which may have values of *dpi::abort-policy::on-warning, dpi::abort-policy::on-error,* or *dpi::abort-policy::struggle-on* (see 15.6).

If a structure warning occurs in a BLOCK whose *abort-policy* presentation parameter has a value of *dpi::abort-policy::on-error*, it typically does not cause a structure exception to occur. Instead, it causes a warning message to be issued to the print requestor in an implementation-dependent way.

If a structure warning occurs in a BLOCK whose *abort-policy* presentation parameter has a value of *dpi::abort-policy::on-warning*, a structure exception is raised in addition to a warning message being sent to the print requestor.

The following subclauses describe the handling of structure exceptions.

36.2.1 dpi::abort-policy::struggle-on

If a structure exception is raised in a BLOCK for which the *abort-policy* presentation parameter value is *dpi::abort-policy::struggle-on*, then the exception is said to be *handled*. An implemetation dependent warning message to the print requestor may be issued.

NOTE 218 This warning message might be transmitted to the process that invoked the SPDL process by a reverse channel whose structure and protocols are not defined by this International Standard, or it might be printed on a break page or banner page.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642804 128

No further processing of the content or structures subordinate to the BLOCK is performed by the presentation process. However, if the BLOCK in which the exception occurs is a PAGE, then the PAGE shall be presented. The page image presented on this page shall be implementation dependent.

Structure processing resumes with the structure immediately following the BLOCK in which the exception occurred. If the exception occurs in a BLOCK subordinate to a PAGE, the cumulative state of the resultant Current Page Image shall be implementation-dependent

NOTE 219 It is recommended that the resultant Current Page Image be as good a representation as possible of the result of the presentation process had the cause of the exception not been present in the document.

36.2.2 dpi::abort-policy::on-error, dpi::abort-policy::on-warning

If a structure exception is raised in a BLOCK for which the *abort-policy* presentation parameter value is *dpi::abort-policy::on-error* or *dpi::abort-policy::on-warning*, then the exception is said to be unhandled. An implemetation-dependent warning message to the print requestor may be issued.

NOTE 220 This warning message might be transmitted to the process that invoked the SPDL process by a reverse channel whose structure and protocols are not defined by this International Standard, or it might be printed on a break page or banner page.

No further processing of the content or structures subordinate to the BLOCK is performed by the presentation process. However, if the BLOCK in which the exception occurs is a PAGE, then the PAGE shall be presented. The page image presented on this page shall be implementation dependent. All pages preceding this page shall be correctly presented, as though the PAGE in which the exception arose had not been present.

When an unhandled exception occurs in a BLOCK, the same structure exception arises in the most immediately superior BLOCK. The effect of that structure exception in the most immediately superior BLOCK will depend on the value of the *abort-policy* presentation parameter in that BLOCK.

If an unhandled structure exception occurs in the DOCUMENT (to which there is no superior BLOCK) then processing of the document terminates. If the DOCUMENT is a single PAGE, then one page image shall be presented, the page image presented on this page shall be implementation dependent.

36.3 Content Exceptions

Content exceptions occur during content processing, as a result of executing any of the following operators: **RaiseException**, and (in some circumstances) **RaiseError** or **RaiseWarning**

The execution of the **RaiseException** operator immediately causes a content exception to occur. This exception may be handled in content (see 36.4) or not. If it is unhandled in content, then a structure exception will be caused. (see 36.5.1).

The execution of the **RaiseError** operator does not itself cause a content exception to occur. Instead, it causes an *error Procedure* to be selected from the dictionary bound to the key *ErrorDict* in *SystemDict*, and the error Procedure is then executed. Error Procedures typically end by executing the **RaiseException** operator, thus causing a content exception to occur. (See 36.6 for complete specification of error Procedures and their uses.)

NOTE 221 *ErrorDict* may be modified in content (see 17.1 6.1). The *error Procedure* bound to a particular interpreter error name may be replaced, for example

The execution of the **RaiseWarning** operator typically does not cause a content exception to occur. Instead, it causes a warning message to be issued to the print requestor in an implementation-dependent way. If, however,the Current Abort Policy component of the Virtual Machine has the value *dpi::abort-policy::on-warning*, then, after issuing the warning message, **RaiseError** shall be invoked with *ContentWarning* as its operand. This may raise an exception (see 36.6).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642805 064

36.4 Content Exception Handling

Content exceptions are the result of executing the **RaiseException** operator. Depending on the context of interpretation of this operator, the exception may be handled or unhandled. If the exception occurs during interpretation of a Procedure called by the **ExecuteTrapped** operator, then the exception will be handled. A Procedure invoked by the **ExecuteTrapped** operator is said to be executed in a *trapped context.* Otherwise, the exception will be unhandled.

36.4.1 Handled exceptions

The **ExecuteTrapped** operator provides a mechanism for executing a Procedure in a context in which content exceptions will be *trapped.* Upon occurrance of an exception, all execution of the Procedure and its subprocedures will halt, and content processing continues with the next object in normal sequence after the **ExecuteTrapped** operator. A boolean true is placed on the Operand Stack to show that an exception was trapped. (If the Procedure had been completely interpreted without an exception arising, a Boolean false would be placed on the Operand Stack.)

The Boolean returned by the **ExecuteTrapped** operator allows it to be followed by an exception handler Procedure, defined by the document, which can be conditionally executed if an exception occurs.

If an exception is trapped and handled in this way, the results are entirely under the control of the document.

36.4.2 Unhandled exceptions

If a content exception occurs in a context without exception trapping, then the content exception is unhandled. The Procedure *ReportErrorInfo* from *ErrorDict* is executed, and then a structure exception in the immediately superior BLOCK occurs.

36.5 Content Exception Operators and Procedures

The specification of these operators and their semantics includes the specification of conditions which may cause content exceptions to be raised as a result of interpretation of the operators. There is a class of generic errors some subset of which may be caused by each operator, which may lead to raising the corresponding generic exception. These generic errors and their semantics are described in 36.6.2. In addition, the individual operator specifications in clause 20 through clause 35 identify operator-specific errors that may lead to exceptions. These errors are also described in 36.6.2.

36.5.1 RaiseException

The **RaiseException** operator takes no arguments and returns no results. It terminates execution of the innermost dynamically enclosing instance of a trapped context, without regard to lexical relationship. The use of trapped contexts is discussed in 36.4.1, and the **ExecuteTrapped** operator itself in 36.5.5 .If there is no such trapped context, then content processing is terminated and status returned to the structure processor shall be *content-warning* or *content-error* as appropriate. These two status values indicate an unhandled content exception.

36.5.2 RaiseError

The **RaiseError** operator takes one operand

<error: Name>

and returns no results. The effect of executing the **RaiseError** operator is to execute the Procedure bound to *error* in *ErrorDict.*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642806 TTO

36.5.3 RaiseWarning

The **RaiseWarning** operator takes one operand

<warning: OctetStringReference>

and returns no results The effect of executing the **RaiseWarning** operator is to execute the *Print* operator with *warning* as an operand, and then conditionally invoke **RaiseError** with *ContentWarning* as its operand. The decision on whether or not to invoke **RaiseError** is based on the Current AbortPolicy of the Virtual Machine; see 36.3 .

36.5.4 Print

The *Print* operator takes one operand

<message: OctetStringReference>

and returns no results. It causes *message* to be issued to the print requestor in an implementation-dependent way.

36.5.5 ExecuteTrapped

The **ExecuteTrapped** operator takes one operand

<P: Procedure>

and returns one result

<trapped: Boolean>

ExecuteTrapped executes the Procedure *P* (as if by the **Execute** operator) If *P* runs to completion normally, **ExecuteTrapped** returns a value of false on the Operand Stack If *P* terminates prematurely as a result of executing the **RaiseException** operator, **ExecuteTrapped** returns the value true on the Operand Stack. Regardless of the outcome, the interpreter resumes execution at the next object in normal sequence after **ExecuteTrapped** operator.

36.5.6 StoreErrorInfo

The *StoreErrorInfo* Procedure in *ErrorDict* takes one operand

<errorname: Name>

and returns one result

<errorname: Name>

This Procedure is executed by the interpreter when an interpreter error occurs; it records information about the error and the context in which it occurred into *ErrorInfoDict* (see 36.7).

The default *StoreErrorInfo* Procedure sets the value of the *newerror* key to true, then sets the values of the *errorname, command, ostack*, and *cstack* keys to the appropriate values (see 36.7). A document may redefine the value of the *StoreErrorInfo* key, but to ensure correct interpretation of subsequent document errors, the new *StoreErrorInfo* Procedure must set the *newerror* key in *ErrorInfoDict* to true as part of its execution, and must ensure that *errorname* remains on the stack following execution of the new *StoreErrorInfo* Procedure.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642807 937

If the default *StoreErrorInfo* Procedure is explicitly executed by a document in content, an interpreter error occurs.

NOTES

222 Due to the desirability of having *StoreErrorInfo* always accessible from the current context of interpretation, an exact copy of the *StoreErrorInfo* key/value pair of *ErrorDict* is maintained by the interpreter in *SystemDict*.

223 If the value of *StoreErrorInfo* is redefined, it is often necessary to redefine the value of the *ReportErrorInfo* Procedure as well (see 36.5.7)

36.5.7 ReportErrorInfo

The *ReportErrorInfo* Procedure in *ErrorDict* takes no operands and returns no results. This Procedure is executed by the interpreter for exceptions unhandled by the document. It sets the value of the *newerror* key in *ErrorInfoDict* to false, then issues an error message to the print requestor in an implementation-dependent manner, based on the information in *ErrorInfoDict* (see 36.7).

A document may redefine the value of *ReportErrorInfo*, but to ensure correct interpretation of subsequent document errors, the new *ReportErrorInfo* Procedure must set the *newerror* key in *ErrorInfoDict* to false as part of its execution.

If the default *ReportErrorInfo* is executed other than from within a content exception handler Procedure, the results are implementation-dependent and may be meaningless, but no interpreter error occurs.

NOTE 224 Due to the desirability of having *ReportErrorInfo* always accessible from the current context of interpretation, an exact copy of the *ReportErrorInfo* key/value pair of *ErrorDict* is maintained by the interpreter in *SystemDict*.

36.6 Interpreter Errors

With the exeception of *Timeout*, Most interpreter errors occur only during the interpretation of an value of type *Operator* in content. However,*Timeout* shall occur only between the execution of objects and *StackOverflow* may occurl when interpreting a value other than an Operator as well as when interpreting an Operator. Interpreter errors may result from a variety of causes including, for example, *divide by zero* and *syntax error*.

When an interpreter error occurs, the Content Processor attempts to restore the state of the Operand Stack to the state which existed immediately prior to interpretation of the Operator which caused the error. The results of this attempt are implementation dependent. Next, it pushes that Operator value on the Operand Stack. Finally, it places the name of the interpreter error (see 36.6.2) on the Operand Stack and executes the **RaiseError** operator. The **RaiseError** operator is executed directly by the interpreter, without searching for a value bound to the name *RaiseError* in the Context Stack.

The subsequent activity of the Content Processor is under control of the error Procedure invoked by the **RaiseError** operator.

36.6.1 Error Procedures and Dictionaries

All of the default error Procedures place the name of the corresponding interpreter error on the Operand Stack, then invoke the *StoreErrorInfo* Procedure from *ErrorDict*, then execute the **RaiseException** operator.

The default contents of *ErrorDict* include the *StoreErrorInfo* Procedure, the *ReportErrorInfo* Procedure, and the set of all error-specific Procedures (see 36.6.2 for a list of all errors). A reference to *ErrorDict* itself is bound to the key *ErrorDict* in *SystemDict*.

The default contents of *ErrorInfoDict*, and their semantics, are described in 36.7. A reference to *ErrorInfoDict* itself is bound to that key *ErrorInfoDict* in *SystemDict*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642808 873

36.6.2 Possible Interpreter Errors

The following is the default list of possible interpreter errors and their semantics. The list of possible interpreter errors on a given implementation may be a superset of this list. The name of the exception raised as a result of the occurence of each error is the same as the name of the underlying error. In the case that by the semantics of this International Standard more than one exception could be raised for a given operator execution, the exception that is actually raised is defined by the implementation.

Throughout this International Standard, the descriptions of operator semantics include specific conditions which cause exceptions during the execution of the operator. There are, in addition, certain generic errors which can cause an exception during the execution of almost any operator. These generic errors are *InvalidAccess*, *LimitCheck*, *NoMemory*, *RangeCheck*, *StackOverflow*, *StackUnderflow*, and *TypeCheck*.

The following is a list of the possible interpreter errors defined by this International Standard:

Table 38

Error	meaning
ContentWarning	**RaiseWarning** operator executed on a Virtual Machine in which the Current Abort Policy has a value of *dpi::abort-policy::on-warning*
ContextStackOverflow	Attempt to **PushContextStack** with a full Context Stack
ContextStackUnderflow	Attempt to **PopContextStack** with no removable Dictionaries on the Context Stack
DataError	Unexpected and illegal characters occur in the *StreamObject* that is input to a filter
InvalidAccess	Attempt to violate an object's access attribute. This error shall occur upon attempting to execute an object with a *NoAcesss* attribute, upon attempting to read an object with an *ExecuteOnly* or *NoAcesss* attribute, or upon attempting to modify an object with a *ReadOnly* , *ExecuteOnly* or *NoAcesss* attribute. Putting an object reference onto the Operand Stack is not an access to the referenced object.
InvalidExit	*Exit* operator executed outside any looping operator
InvalidFont	Invalid operand to **FindFont**
InvalidRestore	Attempt to **RestoreState** when Context Stack or Operand Stack contains ObjectReferences to objects created after the *SaveObject* operand to **RestoreState** was created
IOError	An implementation-dependent error has occurred while reading from a StreamObject
LimitCheck	An implementation limit (one for which there is no specific exception Procedure defined) has been exceeded
NoCurrentPosition	Attempt to execute a path construction operator other than **SetPosition** , **ArcToClockwise** , or **ArcToCounterClockwise** when the current path is the null path
NoMemory	There is not enough memory available to execute the operator successfully
RangeCheck	The value of some operand or the value bound to some key in a dictionary that is an operand of the operator currently being executed is of the correct type, but is not in the allowable range for that operand of that operator. If the operand is a reference to a Vector or Octetstring object into which values are to be placed and the composite object is not of sufficient size to hold all such values, then *RangeCheck* shall occur
StackOverflow	The implementation limit on the depth of the Operand Stack has been exceeded

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642809 70T

Error	meaning
StackUnderflow	Not enough operands on the stack for the operator being executed
SyntaxError	Document content encountered which is not in accordance with the Content Interchange Format (see clause 37 and clause 38)
Timeout	The default implementation time limit for processing a document, or the time limit specified via DPI in the document, has been exceeded
TypeCheck	An operand of the operator currently being executed is of the wrong value type
UndefinedKey	A key given as an operand to the operator currently being executed is not in the Dictionary operand of that operator, in any Dictionary on the Context Stack, or in the enumerated list of Names required by that operator as appropriate to that operator
UndefinedResource	There is no resource of the specified type bound to the INTERNAL RESOURCE IDENTIFIER operand provided to **FindResource**
UndefinedResult	Over/underflow or meaningless result, such as an attempt to *Divide* by 0 or take the **SquareRoot** of a negative number
UnmatchedMark	There is no Mark value on the Operand Stack.

36.7 ErrorInfoDict

The following subclauses describe the key/value pairs in the default *ErrorInfoDict*, and their semantics. Mandatory key/value pairs shall be present in all implementations; optional key/value pairs may be absent in some implementations, in which case default values for the missing key/value pairs are assumed by the exception handling function.

36.7.1 newerror

The mandatory key/value pair *<newerror: Boolean>* is used by the interpreter to track error handling. By default, the value of the key is set to true by the *StoreErrorInfo* Procedure, and is set to false by the *ReportErrorInfo* Procedure. Documents which alter the values of these Procedures must retain the original semantics involving *newerror* (see 36.5.6 and 36.5.7).

36.7.2 errorname

The mandatory key/value pair *<errorname: Name>* is the name of the error that was invoked. A complete list of error names is given in 36.6.2.

36.7.3 command

The mandatory key/value pair *<command: Any>* has as its value the Operator being executed by the interpreter at the time the error occurred. If no Operator was being interpreted at the time the error occured, either its value is the the object being interpreted or, in the case of *TimeOut*, its value may be undefined.

36.7.4 ostack

The mandatory key/value pair *<ostack: VectorReference>* is a reference to a Vector containing a *snapshot* of the entire Operand Stack immediately before the error occurred, stored as if by the **MakeandStoreVector** operator.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642810 421

36.7.5 dstack

The mandatory key/value pair *<dstack: VectorReference>* is a reference to a Vector containing a *snapshot* of the entire Context Stack immediately before the error occurred, stored as if by the **ContextStack** operator.

36.7.6 recordstacks

The optional key/value pair *<recordstacks: Boolean>* controls whether the default *StoreErrorInfo* Procedure (see 36.5.6) is to record the *ostack* and *dstack* snapshots. A value of true means the contents of the stacks are recorded. A value of false means the contents of these stacks are not recorded. If this key/value pair is not present in *ErrorInfoDict*, the default *StoreErrorInfo* Procedure operates as if it were present and had the value true.

36.8 Print requestor

The *print requestor* is an implementation defined channel which is sent information about the presentation of DOCUMENTs, including messages associated with errors and warnings. Whether and how this information is presented is implementation dependent.

NOTE 225 this information may be presented on a series of pages that follow the document and/or sent back to the process that invoked the SPDL process by a reverse channel whose structure and protocols are not defined by this International Standard

37 Clear Text Representation and Interchange Format

The Clear Text Representation and Interchange Format consists of the Clear Text Structure Representation and Interchange Format and the Clear Text Content Representation and Interchange Format.

37.1 Clear Text Structure Representation and Interchange Format

The Clear Text Structure Representation and Interchange Format is based on SGML. The SGML representations of the base structure element value types is given in 37.1.1. The SGML Document Type Definition is given in 4.2.

The SGML Document Type Definition is structured to match as closely as possible the structure of the ASN.1 modules defined in clause 38 . End-tags, except those for elements which have a declared content of "EMPTY", shall not be omitted in the Clear Text Structure Representation and Interchange Format. General entity references are not allowed in content are not allowed in the content of an element in the Clear Text Structure Representation and Interchange Format. The content of the document type declaration subset is limited to general entity declarations which use the external entity specification form for the entity text in the Clear Text Structure Representation and Interchange Format. See ISO/IEC 9070 for details.

NOTE 226 Good SGML style specifies that all inclusions of externally defined document components be identified by entity references which occur at the front of the document. This allows a simple SGML process to find all included document components; this process uses only the SGML information and need not understand the semantics of the what the SGML represents. This International Standard facilitates this aspect of good SGML style, but does not require its use.

37.1.1 Base Structure Element Value Types

The SGML representation of the base structure element value types shall be as follows. Occasionally, additional requirements on the representation of values are specifed in the comments in the SPDL Document Type Definition. These comments are normative

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

37.1.1.1 Boolean type

Values of the *Boolean* type are represented as an attribute of type CDATA of an element. This attribute shall have as its value one of the following instances of character data: "true" or "false".

37.1.1.2 Enumerated types

If the *Enumerated* type consists of a finite set of values specified in this International Standard, then the type is represented as an attribute of an element in which the attribute is a name list that lists representations for the finite set of values. If the *Enumerated* type consists of an open ended set of values specified by Public Object Identifiers, then the type is represented by CDATA with an appropriate NOTATION attribute.

37.1.1.3 Integer type

Values of the *Integer* type are represented as an attribute of an element in which the attribute is a number, if the Integer is positive or a Cardinal, and is a name token which must be of the form of a sequence of decimal digits optionally preceded by a "+" or "-" sign, otherwise.

37.1.1.4 Real type

Values of the *Real* type are represented in the clear text form defined in 37 2.3. Normally, this representation is used for the value of a name token which is an attribute of an element, otherwise this representation is used for the CDATA in an element

37.1.1.5 Printable String type

Values of the *Printable String* type are represented as CDATA which is the content of an element

37.1.1.6 Octet String type

Values of the *Octet String* type are represented as CDATA which is the content of an element. If the Octet String is in a DATA BLOCK, then it is ASCII85 encoded (see 37.2.4). If it is in a LOCAL LOCATION IDENTIFIER, the Octet String is limited to octets in the graphic characters of ISO 646. Otherwise, the Octet String is Hexadecimal encoded.

37.1.1.7 Name type

Values of the *Name* type are represented as CDATA which is the content of an element. Leading and trailing white space characters surrounding the name in the CDATA are ignored.

37.1.1.8 Object Identifier type

Values of the *Object Identifier* type are represented as CDATA which is the content of an element that has a NOTATION attribute with the value "objid". See that NOTATION declaration for the reference to the defining occurance of the notation. Leading and trailing white space characters surrounding the identifier in the CDATA are ignored.

37.1.1.9 Public Identifier type

Values of the *Public Identifier* type are represented as CDATA which is the content of an element that has a NOTATION attribute with the value "pubid". See that NOTATION declaration for the reference to the defining occurance of the notation. Leading and trailing white space characters surrounding the identifier in the CDATA are ignored.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642812 2T4

37.1.1.10 Environment Name type

Values of the *Environment Name* type are represented as CDATA which is the content of an element that has a NOTATION attribute with the value "envnm". See that NOTATION declaration for the reference to the defining occurance of the notation. Leading and trailing white space characters surrounding the name in the CDATA are ignored.

37.1.1.11 Glyph Identifier type

Values of the *Glyph Identifier* type are represented as CDATA which conforms to the syntatic requirements of a structured name in ISO 9541 or a Simple Name as defined in 6.16.5. Leading and trailing white space characters surrounding the identifier in the CDATA are ignored.

37.1.2 SPDL Document Type Definition

In addition to the type representations defined in 37.1.1, when a composite structure element has a subordinate that is a sequence of Cardinals or a similarly simple type, that sequence is encoded as CDATA in which the elements of the sequence are separated by one or more space characters and the individual elements are represented as defined in 37.1.1 . This composite representation is indicated in the comments which are normative.

The SGML Document Type Definition is structured to match as closely as possible the structure of the ASN.1 modules defined in clause 38.

NOTES

227 The names of SGML elements are mostly 7 octets long and always no more than 8 octets. These short names suggest the SPDL structure elements that they represent. SGML Comments are used to better indicate this correspondance. The type of a base structure element is included in parentheses following the name of the structure element in comments.

228 Where there is an enumerated type whose values are in one to one correspondance with the names of structure elements, the same short names are used for both the elements of the enumerated type and for the encoding of the structure elements.

229 Simple base structure elements are often encoded as attributes of the SGML element that represents the composite structure elements to which they are subordinate. The type of the value of that attribute is chosen to be as close to the type of the base structure element as is possible in SGML. Base structure elements that whose type is a union of other types are always encoded as separate elements. There will often be an attribute that indicates which of the types in the union occurs in a given element.

230 If all of the subordinates of a composite structure element are represented as attributes, then the declared content of the SGML element that represents the composite structure element will be "EMPTY". More often, a composite structure element in which the subordinates may occur in any order has some subordinates that are represented as attributes and others that are represented as SGML elements. In such cases, the syntax of SGML requires that the attributes occur before the contained SGML elements. Within each group, however, the subordinates may occur in any order. In a composite structure element in which the subordinates shall occur in the specified order, if the first subordinate can be represented by an attribute, then that is done. All other subordinates will be represented by SGML elements in the content model of the SGML element representing the composite structure element.

231 ASN.1 type definitions are approximated in SGML be defining a parameter entity with the same name as the ASN.1 type (or a shortened form of that name). The value of the parameter entity is the "|" separated group of SGML element names for the structure elements whose value is of that type. This parameter entity is used in the position of the SGML element type in an element declaration that specifies the content model and attributes associated with values of that type. All SGML elements whose value is of that type are defined in one spot and subsequent changes to this International Standard that affect the type definition can be accomplished by editting this single "type definition". (SGML limitations on the number of element names that can occur in an name group may require defining more than one parameter entity when there are many names in the group)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642813 130

The SGML Document Type Definition:

```
<!-- (C) International Organization for Standardization 1993
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879-1986, provided this notice is included in all copies.
     Creator:  Stephen Zilles
     Version: 1.03  94-Dec-16
-->
<!-- Public document type definition. Typical invocation:
        <!DOCTYPE SPDL PUBLIC "ISO/IEC 10180//DTD Standard Page
                Description Language//EN"                          -->
<!ENTITY % doctype "spdl"    -- Document type generic identifier -->
<!--This is a document type definition for ISO 10180, the Standardized
    Page Description Language. It follows as closely as possible the
    structure of the corresponding Binary Representation and
    Interchange Format in Clause 38 of this International Standard.
    The definition is divided into "modules" each of which defines
    the ENTITIES used, the ELEMENTs defined and ATTLISTs for those
    ELEMENTs.
                                                                   -->
                  <!-- included DTD Entities -->
<!ENTITY % fontset PUBLIC "ISO 9541-2:1991//DTD Font Attribute Set//EN">
<!ENTITY % strucnm PUBLIC "ISO 9541-2:1991//DTD Structured Name//EN">
<!-- NOTE that corrigenda have been issued against this standard and
         should be incorporated in the public text for these DTDs   -->
%fontset;
%strucnm;
               <!-- Structure Definition "module" -->
<!-- Notations used for Identifiers                                -->
<!NOTATION pubid           PUBLIC
           "ISO/IEC 9070//NOTATION Public Identifiers//EN"          >
<!NOTATION objid           PUBLIC
           "ISO/IEC 10180//NOTATION Object Identifier type//EN"     >
<!NOTATION envnm           PUBLIC
           "ISO/IEC 9070//NOTATION object-name component//EN"       >
<!-- Types for Base Structure Elements                             -->
<!-- Entities                                                      -->
<!ENTITY % Boolean  "CDATA"                          -- Boolean type -->
<!--     The only allowed values are "true" or "false", in lowercase#-->
<!ENTITY % Integer  "CDATA"                          -- Integer type -->
<!ENTITY % Cardinl  "NUMBER"                        -- Cardinal type -->
<!ENTITY % PosIntg  "NUMBER"              -- Postitive Integer type -->
<!ENTITY % Real     "CDATA"                             -- Real type -->
<!ENTITY % Number   "CDATA"                          -- Number type -->
<!ENTITY % NnNgNum  "CDATA"              -- Non Negative Number type -->
<!ENTITY % Name     "CDATA"                             -- Name type -->
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642814 077


```
<!ENTITY % PrntStr              "CDATA"                  -- Printable String type -->
<!ENTITY % PblcIdn              "CDATA"                -- Public Identifier type -->
<!-- the environment identifier type is defined in three pieces to
     avoid overrunning the GRPCNT quantity limit                                  -->
<!ENTITY % envid
  "hintnm | nSPDLnm | envrsid | fnidspc | ndxfnid | gmap | fntrsid"
                                    -- environment identifier type -->
<!ENTITY % envid1
    "namddim | namdxyd | namdloc | mednam | medmclr | medmtyp | medtooh |
     medgran | edgfnsh | medmid | fnshnam | stchnam | stchtyp | bindnam |
     bindtyp | bindclr | trimnam | diecnam | diecnm | pnchnam | perfnam"
                          -- environment identifier type, continued -->
<!ENTITY % envid2
    "perftyp | slitnam | nsrtnam | nsrtnm | covrnam | covrnm | foldnam |
     nfinprm | colrsid | colrid | strpage | seppage | endpage | abrtdpi |
     outbnam"
                          -- environment identifier type, continued -->
<!ELEMENT (%envid;)   - -  CDATA          -- environment identifier   -->
<!ATTLIST (%envid;)
                          notation NOTATION (pubid | objid | envnm)
                                            #REQUIRED
         -- notation in which environment identifier is represented -->
<!ELEMENT (%envid1;)  - -  CDATA          -- environment identifier   -->
<!ATTLIST (%envid1;)
                          notation NOTATION (pubid | objid | envnm)
                                            #REQUIRED
         -- notation in which environment identifier is represented -->
<!ELEMENT (%envid2;)  - -  CDATA          -- environment identifier   -->
<!ATTLIST (%envid2;)
                          notation NOTATION (pubid | objid | envnm)
                                            #REQUIRED
         -- notation in which environment identifier is represented -->
<!-- the public object identifier type                                           -->
<!ENTITY % pubobid
          "pubobid | clrsnm | psetid | pcolrid | matchrl | satisfc |
           dpiname | propnam | plexdpi | finopnm | colrcls | ospcnm"
                                  -- public object identifier type -->
<!ELEMENT (%pubobid;) - -  CDATA        -- public object identifier   -->
<!ATTLIST (%pubobid;)
                          notation NOTATION (pubid | objid)
                                            #REQUIRED
       -- notation in which public object identifier is represented -->
<!-- Structure Elements, SPDL Instance and TokenSequences                        -->
<!-- Entities                                                                    -->
<!ENTITY % documnt  "pageset|picture"                       -- Document -- >
<!-- the Structure Element PAGE is intentionally not encoded                     -->
<!ENTITY % ifntspc
      "fntrspc|fnt1spc|rfntspc|cfntspc|ufntspc"
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                                    -- includable  font specification -->
<!ENTITY % fontspc  "fnidspc|%ifntspc;"        -- font specification -- >
<!ENTITY % inclst0
  "pageset | picture"
                                    -- Includable Structures, part 0-->
<!ENTITY % inclst1
  "pictbdy | nonSPDL | prologue |infrdcl | hint |
cntxdcl | resdefn
   | stupprc | tknseqn | fontrfr| dictspc | %ifntspc;"
                                    -- Includable Structures, part 1 -->
<!ENTITY % inclst2
  " fontset |gmapspc | gmapspc | fndxmap | gndxtbl | datsspc
    | clrsspc | patnspc | formspc | dpidcls | addldpi"
                                    -- Includable Structures, part 2 -->
<!-- The list of Includable Structure Elements:
        pageset   pageset
        picture   picture
        pictbdy   picture body
        nonSPDL   non-SPDL picture body
        prologue  prologue
        infrdcl   informative declaration
        hint      hint
        cntxdcl   context declaration
        resdefn   resource definition
        stupprc   setup procedure
        tknseqn   tokensequence
        fontrfr   font reference
        dictspc   dictionary resource specification
        fontspc   font specification
        fntrspc   font reference font specification
        fntlspc   fonttype1 specification
        rfntspc   remapped font specification
        cfntspc   composite font specification
        ufntspc   user font specification
        gmapspc   glyph index map specification
        fndxmap   font index map
        fontset   9541 font attribute set
        gndxtbl   glyph index map translation table
        datsspc   data source specification
        clrsspc   color space resource specification
        patnspc   pattern resource specification
        formspc   form resource specification
        dpidcls   dpi declaration
        addldpi   additional dpi                                    -->
<!ENTITY % tknsref  "(tknseqn | strctid)"    -- tokensequence-or-ref -->
<!-- Elements                                                       -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                     -->
<!ELEMENT %doctype;   - - (%documnt;|envres|%inclst1;|%inclst2;)
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642816 94T


```
                        +( comment )
                                          --SPDL top level structure -->
<!ELEMENT tknseqn     - -  CDATA                    -- tokensequence -->
<!ELEMENT comment     - -  CDATA                     -- SPDL comment -->
<!-- Comment and Public Identifiers                                   -->
<!--   Comments are encoded using the SGML comment convention         -->
<!--   The "Object" is identifier by the SPDL DTD reference           -->
<!--   The prefix on Public Identifiers defined by this standard      -->
<!--   is "ISO/IEC 10180//"                                           -->
                    <!-- SPDL Document  "module" -->
<!-- High Level Structure Elements                                    -->
<!-- Entities                                                         -->
<!ENTITY % pgstbdy  "(pageset|picture|strctid)*"     -- Pageset Body -->
<!ENTITY % body     "(picture|tknseqn|strctid)*"  -- body-of-picture -->
<!ENTITY % prlgref  "(prologue|strctid)"          -- prologue-or-ref -- >
<!-- Elements                                                         -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                       -->
<!ELEMENT pageset     - -  ((%prlgref;)?,%pgstbdy;) >
<!ELEMENT picture     - -  (%body; | nonSPDL | strctid) >
<!ELEMENT pictbdy     - -  ((%prlgref;)?,%body;)     -- Picture Body -->
<!ELEMENT nonSPDL     - -  CDATA            -- non-SPDL picture body -->
<!-- Attributes of Elements                                           -->
<!--      ELEMENTS         NAME     VALUE     DEFAULT                 -->
<!ATTLIST picture
                         contrep  CDATA     #REQUIRED
                                        -- content representation -->
<!-- value must be a Public Identifier as defined in ISO/IEC 9070    -->
<!-- see annex B for values defined in this standard        -->
<!-- for non-SPDL pictures, it specifies the notation used           -->
<!ATTLIST nonSPDL
                         encoded  (true|false) false
             -- indication whether ASCII85 encoding is used on data -->
                  <!-- Prologue Structure Elements -->
<!-- Entities                                                         -->
<!ENTITY % infrref  "(infrdcl | strctid)"
                                        -- informative-decl-or-ref -->
<!ENTITY % hintref  "(hint | strctid)"              -- hint-or-ref -- >
<!ENTITY % dpirref  "(dpidecl | strctid)"  -- dpi declaration-or-ref -- >
<!ENTITY % cntxref  "(cntxdcl | strctid)"
                                    -- Context Declaration-or-ref -->
<!ENTITY % resdref  "(resdefn | resdecl |cntxadd | strctid)"
                       -- resource-definition-or declaration-or-ref -->
<!ENTITY % stupref  "(stupprc | strctid)"  -- setup-procedure-or-ref -- >
<!-- Elements                                                         -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                       -->
<!ELEMENT prologue    - -  ( extndcl*, (%infrref;)?, nSPDLop*,
                             dpidcls?, (%cntxref;)?, (%resdref;)*,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                              (%stupref;)? )                                >
<!-- where no explicit elements are defined for the following
     Structure Elements:
          extndcl*      external-declaration-list
          (%hintref;)*  informative declarations
          nSPDLop*      non-SPDL operation-list
          (%cntxref;)?  context-declaration
          (%resdref;)*  resource defn, resource decl, or context addition
          (%stupref;)?  setup-procedure                                  -->
<!ELEMENT dpidcls     - -  (%dpirref;)*      -- dpi-declaration-list -->
<!ELEMENT infrdcl     - -  (%hintref;)*   -- informative declarations-->
<!ELEMENT hint        - -  (hintnm,hintval)
                                        -- hint-name-and-hint-value -->
<!-- hintnm = envid(see base types above)                  hint-name -->
<!ELEMENT hintval     - -  ANY                          -- hint-value -->
<!-- the type of the content(hintval) depends on the hintname(hintnm)-->
<!ELEMENT nSPDLop     - -  (nSPDLnm,nSPDLvl)
                                    -- non-SPDL-operation-and-value -->
<!-- nSPDLnm = envid(see base types above)   non-SPDL-operation-name -->
<!ELEMENT nSPDLvl     - -  ANY           -- non-SPDL-operation-value -->
<!-- the type of the content (nSPDLvl) depends on the opname(nSPDLnm)-->
<!ELEMENT cntxdcl     - -  (intrsid)*-- context-dictionary-name-list -->
<!-- each intrsid must be the name of a context-dictionary          -->
<!ELEMENT cntxadd     - -  (intrsid)      -- context-dictionary-name -->
<!ELEMENT stupprc     - -  %tknsref;-- setup-procedure-tokensequence -->
                    <!-- SPDL External Reference Module -->
<!-- Assignment List                                                -->
<!-- Entities                                                       -->
<!ENTITY % locatid  "pubobid|loclcid|sgmlext|sgmlent"
                                          -- location identifier -->
<!-- where the alternatives for location identifiers are:
          pubobid       public object identifier
          loclcid       local location identifier
          sgmlext       encoding dependent location identifier-1
          sgmlent       encoding dependent location identifier-2     -->
<!-- Elements                                                       -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                     -->
<!ELEMENT extndcl     - -  (strctid, (%locatid;))
                                        -- external declaration -->
<!ELEMENT strctid     - -  CDATA     -- structure element identifier -->
<!-- pubobid = pubobid(see ENTITY above)      sgml public identifier -->
<!ELEMENT loclcid     - -  CDATA
                    -- local location identifier (Octet/Char String) -->
<!ELEMENT sgmlext     - -  CDATA
               -- sgml external identifier (external id per ISO 8879) -->
<!ELEMENT sgmlent     - O  EMPTY
 -- sgml external identifier as specifed via an ISO 8879 entity name -->
<!-- Attributes of Elements                                         -->
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<!--      ELEMENTS          NAME      VALUE      DEFAULT                -->
<!ATTLIST sgmlent
                            entname   ENTITY     #REQUIRED
           -- name of entity whose declaration defines data location -->
<!ATTLIST extndcl
                            strclid   (%inclst0;|%inclst1;
                                       |%inclst2;|fontspc)
                                                 #REQUIRED
             -- structure element class identifier (enumerated type) -->
<!-- NOTE: the class identifiers have been chosen to be the same as
           the generic identifiers for the elements they correspond to
           except for font specification (fontspc) which is not encoded
           as a SGML element because it is a choice structure element.
                                                                       -->
              <!-- SPDL Resource Definition "module" -->
<!-- Assignment List                                                  -->
<!-- Entities                                                         -->
<!-- fontspc: font spec (see above in includable instances)           -->
<!ENTITY % resspec
    "dictspc|%fontspc;|gmapspc|clrsspc|datsspc|patnspc|formspc"
                                          -- resource specification -->
<!ENTITY % resclas
    "(Dict|Font|Encoding|ColorSp|DataSrc|Pattern|Form|Filter)"
                               -- resource classes (enumerated type) -->
<!-- Elements                                                         -->
<!--      ELEMENTS     MIN  CONTENT (EXCEPTIONS)                      -->
<!ELEMENT envres       - -  ( extndcl*, (%infrref;)*,
                              (resdecl | cntxadd)*, (resdefn | resundf))
                                          -- environment resource -->
<!ELEMENT resdefn      - -  (envrsid,((%resspec;)|strctid))
                                           -- resource definition -->
<!ELEMENT resundf      - -  (envrsid)        -- resource undefinition -->
<!ELEMENT resdecl      - -  (intrsid,envrsid) -- resource declaration -->
<!ELEMENT intrsid      - -  CDATA
                              -- internal resource identifier (name) -->
<!-- envrsid = envid(see types above) environment resource identfier -->
<!-- Attributes of Elements                                           -->
<!--      ELEMENTS          NAME      VALUE      DEFAULT                -->
<!ATTLIST resdefn
                            resclid   %resclas; #REQUIRED
                       -- resource class identifier (enumerated type) -->
<!ATTLIST resundf
                            resclid   %resclas; #REQUIRED
                       -- resource class identifier (enumerated type) -->
<!ATTLIST resdecl
                            resclid   %resclas; #REQUIRED
                       -- resource class identifier (enumerated type) -->
<!-- Resource Specifications                                          -->
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642819 659


```
<!-- Entities                                                          -->
<!ENTITY % psetspc  "(psetid | psetlst)?"-- primary set specification-->
<!-- Elements                                                          -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                        -->
<!ELEMENT dictspc     - -  (%tknsref;)+
                           -- dictionary resource specification -->
<!ELEMENT datsspc     - -  (%locatid;|datablk)
                           -- data source resource specification -->
<!ELEMENT datablk     - -  CDATA        -- data block (octet string)-->
<!--                the CDATA shall be an ASCII85 encoded octetstring -->
<!ELEMENT clrsspc     - -  (clrsnm, %psetspc;, (%tknsref;)+)
                           -- color space resource specification -->
<!-- clrsnm = pubobid(see types above) color space family identifier -->
<!-- psetid = pubobid(see types above)        primary set identifier -->
<!ELEMENT psetlst     - -  (pcolrid+)            -- primary set list -->
<!-- pcolrid = pubobid(see types above)      primary color identifer -->
<!ELEMENT patnspc     - -  (%tknsref;)+
                          -- pattern resource specification -->
<!ELEMENT formspc     - -  (%tknsref;)+
                             -- form resource specification -->
<!-- Attributes of Elements                                            -->
<!--      ELEMENTS         NAME     VALUE     DEFAULT                  -->
          <!-- SPDL Fonts and Glyph Index Map  "module" -->
<!-- Indexed Font Resource Specifications                              -->
<!-- Entities                                                          -->
<!-- fontspc: font spec (see above in includable instances)            -->
<!ENTITY % fmaplst  "escchar? & shftout? & shftin? & subvect?"
                                  -- fmaptype parameters list -->
<!ENTITY % fontlst  "((%fontspc;)|strctid)+"     -- font-or-ref list -- >
<!ENTITY % gndxref  "gndxtbl|strctid" -- glyph-indx-xlate-tbl-or-ref -- >
<!ENTITY % fndxref  "fndxmap | strctid"   -- font-index-map-or-ref -->
<!ENTITY % fnrfref  "(fontrfr|strctid)"     -- font-reference-or-ref -- >
<!ENTITY % remap    "(gmap | (%gndxref;))"                  -- remap -- >
<!-- Elements                                                          -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                        -->
<!-- fnidspc = envid(see above)   font identifier font specification -->
<!ELEMENT fntrspc     - -  (%fnrfref;, gmap)
                           -- font reference font specification -->
<!ELEMENT fntlspc     - -  CDATA     -- fonttype1 font specification -->
<!ELEMENT rfntspc     - -  (ndxfnid, (%remap;)?)
                           -- remapped font specification -->
<!-- ndxfnid = envid(see base types above)   indexed font identifier -->
<!-- gmap = envid(see base types above)   glyph index map identifier -->
<!ELEMENT gndxtbl     - -  CDATA      -- glyph index map translation -->
<!-- The CDATA is a list of Cardinals separated by one or more blanks
     The value of the "size" attribute of this element specifies
     exactly how many Cardinals shall be in this list.                -->
<!ELEMENT cfntspc     - -  ((%fmaplst;),(%fndxref;),(%fontlst;))
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                                    -- composite font specification -->
<!ELEMENT escchar      - -  CDATA                -- escchar (cardinal) -->
<!ELEMENT shftout      - -  CDATA               -- shiftout (cardinal) -->
<!ELEMENT shftin       - -  CDATA                -- shiftin (cardinal) -->
<!ELEMENT subvect      - -  CDATA          -- subvector (octet string) -->
<!ELEMENT fndxmap      - -  CDATA                     -- font index map -->
<!-- The CDATA is a list of Cardinals separated by one or more blanks
     The value of the "size" attribute of this element specifies
     exactly how many Cardinals shall be in this list.                  -->
<!ELEMENT ufntspc      - -  (%tknsref;)+   -- user font specification -->
<!-- Attributes of Elements                                             -->
<!--       ELEMENTS          NAME      VALUE      DEFAULT               -->
<!ATTLIST gndxtbl
                            size      NUMBER     #REQUIRED
                                               -- table size (cardinal) -->
<!ATTLIST cfntspc
                            type      NUMBER     #REQUIRED
                                               -- fmaptype (cardinal) -->
<!ATTLIST fndxmap
                            size      NUMBER     #REQUIRED
                                               -- map size (cardinal) -->
<!-- Font Reference                                                     -->
<!-- Entities                                                           -->
<!ENTITY % fntaref  "fontset|strctid"      -- font attribute set-or-ref -->
<!-- Elements                                                           -->
<!--       ELEMENTS     MIN  CONTENT (EXCEPTIONS)                       -->
<!ELEMENT fontrfr      - -  (fntrsid?, reqprop?, advprop?, matchrl?,
                             satisfc?)                -- font reference -->
<!-- fntrsid = envid(see base types above)  font resource identifier -->
<!ELEMENT reqprop      - -  (%fntaref;)          -- required properties -->
<!ELEMENT advprop      - -  (%fntaref;)          -- advisory properties -->
<!-- matchrl = pubobid(see above)       match rules(enumerated type) -->
<!-- satisfc = pubobid      satisfaction criterion (enumerated type) -->
<!-- Glyph Map Specifications                                           -->
<!-- Entities                                                           -->
<!ENTITY % glyphid  "smplname|strucnm"
                                -- Glyph Identifer Type alternatives -->
<!-- Elements                                                           -->
<!--       ELEMENTS     MIN  CONTENT (EXCEPTIONS)                       -->
<!ELEMENT gmapspc      - -  (%glyphid;)+
                            -- glyph index map resource specification -->
<!ELEMENT smplname     - -  CDATA        -- name local to the document -->
<!-- Attributes of Elements                                             -->
<!--       ELEMENTS          NAME      VALUE      DEFAULT               -->
<!ATTLIST gmapspc
                            size      NUMBER     #REQUIRED
                                               -- map size (cardinal) -->
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

© ISO/IEC

```
      <!-- SPDL Document Production Instructions  "module" -->
<!-- Useful Data Types: dimensions                                        -->
<!-- Entities                                                             -->
<!ENTITY % dimensn  "(pnchdia | pnchoff | trimoff | procoff)"
                                              -- dimensions type -->
<!ENTITY % dimsval  "(numrdim | namddim)"       -- dimension value -->
<!-- Elements                                                             -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                           -->
<!ELEMENT %dimensn;   - -  %dimsval;                  -- dimensions -->
<!ELEMENT numrdim     - -  CDATA
                        -- numeric dimension (non-negative number) -->
<!-- namddim = envid(see base types above)          named dimension -->
<!-- Attributes of Elements                                               -->
<!--      ELEMENTS         NAME     VALUE     DEFAULT                    -->
<!ATTLIST %dimensn;
                         tolrnce  NMTOKEN   #IMPLIED
                               -- tolerance (non-negative number) -->
<!-- Useful Data Types: xydimensions                                      -->
<!-- Entities                                                             -->
<!ENTITY % xydimsn  "(medmsz | refrsz | trimdim | diecpos)"
                                            -- xydimensions type -->
<!ENTITY % xydmval  "(numrxyd | namdxyd)"     -- xydimensions value -- >
<!-- Elements                                                             -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                           -->
<!ELEMENT %xydimsn;   - -  %xydmval;                -- xydimensions -->
<!ELEMENT numrxyd     - O  EMPTY             -- numeric xydimensions -->
<!-- namdxyd = envid(see base types above)       named xydimensions -->
<!-- Attributes of Elements                                               -->
<!--      ELEMENTS         NAME     VALUE     DEFAULT                    -->
<!ATTLIST %xydimsn;
                         tolrnce  NMTOKEN   #IMPLIED
                               -- tolerance (non-negative number) -->
<!ATTLIST numrxyd
                         xdim     NMTOKEN   #REQUIRED
                              -- x-dimension (non-negative number) --
                         ydim     NMTOKEN   #REQUIRED
                              -- y-dimension (non-negative number) -->
<!-- Useful Data Types: head locations                                    -->
<!-- Entities                                                             -->
<!ENTITY % hdlcval  "(numrloc | namdloc)"    -- head locations value -->
<!-- Elements                                                             -->
<!--      ELEMENTS    MIN  CONTENT (EXCEPTIONS)                           -->
<!ELEMENT headloc     - -  %hdlcval;               -- head locations -->
<!ELEMENT numrloc     - -  (hdloctn)+       -- numeric head location -->
<!ELEMENT hdloctn     - -  CDATA
                           -- head location (non-negative number) -->
<!-- namdloc = envid(see base types above)      named head locations -->
<!-- Attributes of Elements                                               -->
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642822 143


```
<!--      ELEMENTS          NAME      VALUE      DEFAULT                  -->
<!ATTLIST headloc
                            tolrnce   NMTOKEN    #IMPLIED
                                  -- tolerance (non-negative number) -->
<!-- Useful Data Types: edge                                             -->
<!-- Entities                                                            -->
<!ENTITY % edge      "edge      (topedge | botedge | lftedge | rgtedge)
                                        #REQUIRED"       -- edge type -->
<!-- This parameter entity is used as shown below:                       --
--ELEMENT pnchedg      - O  EMPTY  --                    -- punch edge --
--ATTLIST pnchedg
                            %edge; --                           -- edge -->
<!-- DPI Declaration                                                     -->
<!ELEMENT dpidecl      - -  ( meddpi? & medsdpi? & cmeddpi? & colrdpi?
                            & copidpi? & pagedpi? & spagdpi? & plexdpi?
                            & sidedpi? & xshfdpi? & yshfdpi? & csiddpi?
                            & fnshdpi? & auxpdpi? & outbdpi? & dcmtdpi?
                            & dstrdpi? & denddpi? & timedpi? & abrtdpi?
                            & ospcdpi? & addldpi* )
                                              -- dpi declaration -- >
<!-- The list of DPI Structure Elements:
        medsdpi   medium select dpi
        meddpi    medium dpi
        cmeddpi   current medium dpi
        colrdpi   current colorant set dpi
        copidpi   copies dpi
        pagedpi   page select dpi
        plexdpi   plex dpi
        sidedpi   sides dpi
        xshfdpi   x-image shift dpi
        yshfdpi   y-image shift dpi
        csiddpi   current side dpi
        fnshdpi   finishing dpi
        auxpdpi   auxilliary page type dpi
        outbdpi   output bin dpi
        ospcdpi   output specification dpi
        dcmtdpi   document comment dpi
        dstrdpi   document start message dpi
        denddpi   document end message dpi
        timedpi   timeout dpi
        abrtdpi   abort policy dpi
        addldpi   additional dpi                                        -->
<!-- Additional DPI                                                     -->
<!ELEMENT addldpi      - -  (dpiname, dpivalu)       -- additional dpi -->
<!-- dpiname = pubobid(see base types above)     additional dpi name -->
<!ELEMENT dpivalu      - -  ANY                 -- additional dpi value -->
<!-- the type of the content (value) depends on the dpiname             -->
<!-- List of suffixs commonly used for DPI related elements
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        clr       color (also colr)
        cnt       count
        col       column
        dia       diameter
        dim       dimension
        edg       edge (also edge)
        id        identifier
        lng       length
        loc       location
        msg       message
        nam       name (also name and nm)
        num       number
        off       offset
        pag       page (also page)
        pos       position
        prm       parameters
        prp       property
        shf       shift
        spc       specification
        sz        size
        typ       type (also type)
        wgt       weight
        xyd       xydimension                                          -->
<!-- Medium Document Production Instructions                           -->
<!ELEMENT meddpi      - -  (meddecl*)                  -- medium dpi -->
<!ELEMENT meddecl     - -  (medspc)            -- medium declaration -->
<!ATTLIST meddecl
                           medid    CDATA     #REQUIRED
                                    -- medium identifier (name) -->
<!ELEMENT medspc      - -  (mednam?, medmsg?, medprp?)
                                    -- medium specification -->
<!-- mednam = envid(see base types above)                 medium name -->
<!ELEMENT medmsg      - -  CDATA -- medium message (printable string)-->
<!ELEMENT medprp      - -  ( medmsz? & medmclr? & medmwgt? & medmtyp?
                           & medtooh? & medgran? & medpnch? & medmult?
                           & medfedg? & medlbls? & addlprp* )
                                    -- medium properties -->
<!-- The list of Medium Properties Structure Elements:
        medmsz    medium size
        medmclr   medium color
        medmwgt   medium weight
        medmtyp   medium type
        medtooh   medium tooth
        medgran   medium grain
        medpnch   medium pre-punch
        medmult   medium multi-part
        medfedg   medium finish edge
        medlbls   medium labels
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        addlprp   additional properties                                  -->
<!-- medium size = xydimension(see useful types above)                   -->
<!-- medmclr = envid(see base types above)                 medium color -->
<!ELEMENT medmwgt     - O  EMPTY        -- medium weight (any order) -->
<!ATTLIST medmwgt
                          value    NUMBER    #REQUIRED
                                  -- medium weight value (cardinal) --
                          tolrnce  NUMBER    #IMPLIED
                              -- medium weight tolerance (cardinal) -->
<!-- medmtyp = envid(see base types above)                 medium type -->
<!-- medtooh = envid(see base types above)                medium tooth -->
<!-- medgran = envid(see base types above)                medium grain -->
<!ELEMENT medpnch     - -  (pnchdia? & pnchoff? & headloc?)
                             -- medium pre-punch (any order) -->
<!ATTLIST medpnch
                          count    NUMBER    #REQUIRED
                                         -- punch count (cardinal) --
                          %edge;    -- edge(see useful types above) -->
<!-- punch diameter = dimensions(see useful types above)                 -->
<!-- punch offset = dimensions(see useful types above)                   -->
<!-- punch location = head locations(see useful types above)             -->
<!ELEMENT medmult     - O  EMPTY    -- medium multi-part (any order) -->
<!ATTLIST medmult
                          seqnnum  NUMBER    #REQUIRED
                                     -- sequence number (cardinal) --
                          seqnlng  NUMBER    #REQUIRED
                                     -- sequence length (cardinal) -->
<!ELEMENT medfedg     - -  (edgfnsh?)        -- medium finished edge -->
<!ATTLIST medfedg
                          %edge; -- edge id(see useful types above) -->
<!-- edgfnsh = envid(see base types above)           edge finish type -->
<!ELEMENT medlbls     - O  EMPTY                     -- medium labels -->
<!ATTLIST medlbls
                          percolm  NUMBER    #REQUIRED
                                          -- per column (cardinal) --
                          perrow   NUMBER    #REQUIRED
                                           -- per row (cardinal) -->
<!ELEMENT addlprp     - -  (propnam, propval)-- additional properties-->
<!-- propnam = pubobid(see base types above)            property name -->
<!ELEMENT propval     - -  ANY
                         -- property value (type dependent on name) -->
<!ELEMENT medsdpi     - -  (medslct)*           -- medium select dpi -->
<!ELEMENT medslct     - -  (medmid)              -- medium selection -->
<!ATTLIST medslct
                          start    NUMBER    #REQUIRED
                  -- start page = page identifier (positive integer) --
                          end      NUMBER    #REQUIRED
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                    -- end page = page identifier (positive integer) -->
<!-- medmid = envid(see base types above)          medium identifier -->
<!ELEMENT cmeddpi     - -  CDATA        -- current medium dpi (name) -->
<!-- Presentation DPI                                                 -->
<!ELEMENT copidpi     - O  EMPTY >
<!ATTLIST copidpi
                           copies   NUMBER    #REQUIRED
                                          -- copies dpi (cardinal) -->
<!ELEMENT pagedpi     - -  (pagslct)+             -- page select dpi -->
<!ELEMENT pagslct     - O  EMPTY                   -- page selection -->
<!ATTLIST pagslct
                           start    NUMBER    #REQUIRED
                  -- start page = page identifier (positive integer) --
                           end      NUMBER    #REQUIRED
                    -- end page = page identifier (positive integer) -->
<!ELEMENT spagdpi     - -  (pagslct)+
                                   -- supplementary page select dpi -->
<!ELEMENT sidedpi     - O  EMPTY >
<!ATTLIST sidedpi
                           sides    NUMBER    #REQUIRED
                      -- sides dpi (cardinal, shall be "1" or "2") -->
<!-- plexdpi = pubobid(see base types above)               plex dpi -->
<!ELEMENT xshfdpi     - O  EMPTY >
<!ATTLIST xshfdpi
                           shift    NMTOKEN   #REQUIRED
                              -- x-image shift dpi (number) -->
<!ELEMENT yshfdpi     - O  EMPTY >
<!ATTLIST yshfdpi
                           shift    NMTOKEN   #REQUIRED
                              -- y-image shift dpi (number) -->
<!ELEMENT csiddpi     - O  EMPTY >
<!ATTLIST csiddpi
                           side     NUMBER    #REQUIRED
                -- current side dpi (cardinal, shall be "1" or "2") -->
<!-- Finshing DPI                                                    -->
<!-- Useful Finishing Types                                          -->
<!ELEMENT refredg     - O  EMPTY >
<!ATTLIST refredg
                           %edge;
                                        -- reference edge = edge -->
<!ELEMENT jogedge     - O  EMPTY >
<!ATTLIST jogedge
                           %edge;
                                             -- jog edge = edge -->
<!ENTITY % fnshspc  "( fnshnam | fspclst )"
                                     -- finishing specification -->
<!-- The list of finishing process specification structure elements
        stchspc   stitching specification
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        bindspc   binding specification
        trimspc   trimming specification
        diecspc   die cutting specification
        pnchspc   punching specification
        perfspc   perforating specification
        slitspc   slitting specification
        nsrtspc   insert specification
        covrspc   covers specification
        foldspc   folding specification
        ofinspc   other finishing specification                          -->
<!ENTITY % stchspc  "(stchnam | stchprm)"
                                          -- stitching specification -->
<!ENTITY % bindspc  "(bindnam | bindprm)"
                                            -- binding specification -->
<!ENTITY % trimspc  "(trimnam | trimprm)"
                                            -- triming specification -->
<!ENTITY % diecspc  "(diecnam | diecprm)"
                                        -- die cutting specification -->
<!ENTITY % pnchspc  "(pnchnam | pnchprm)"
                                            -- punching specification -->
<!ENTITY % perfspc  "(perfnam | perfprm)"
                                        -- perforating specification -->
<!ENTITY % slitspc  "(slitnam | slitprm)"
                                            -- slitting specification -->
<!ENTITY % nsrtspc  "(nsrtnam | nsrtprm)"
                                             -- insert specification -->
<!ENTITY % covrspc  "(covrnam | covrprm)"
                                            -- covers specification -->
<!ENTITY % foldspc  "(foldnam | foldprm)"
                                           -- folding specification -->
<!ENTITY % fprcsp1  "( %stchspc; | %bindspc; | %trimspc; | %diecspc; )"
                     -- finishing process specification, part 1 -->
<!ENTITY % fprcsp2  "( %pnchspc; | %perfspc; | %slitspc; | %nsrtspc;
                    | %covrspc; | %foldspc; | ofinspc )"
                     -- finishing process specification, part 2 -->
<!ELEMENT fnshdpi     - -  (fnshmsg?, %fnshspc;)    -- finishing dpi -->
<!ELEMENT fnshmsg     - -  CDATA
                       -- finishing message (printable string) -->
<!-- fnshnam = envid(see base types above)          named finishing -->
<!ELEMENT fspclst     - -  (%fprcsp1; | %fprcsp2;)+
                                    -- finishing specification list -->
<!-- stchnam = envid(see base types above)          named stitching -->
<!ELEMENT stchprm     - -  ( refrsz? & refredg? & jogedge? & procoff?
                           & headloc? & stchtyp? )
                                 -- stitching parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                     -->
<!-- procoff = dimension(see useful types above)                      -->
<!-- headloc(see useful types above)                                  -->
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<!-- stchtyp = envid(see base types above)                stitch type -->
<!-- bindnam = envid(see base types above)              named binding -->
<!ELEMENT bindprm     - -  ( refrsz? & refredg? & jogedge? & bindtyp?
                           & bindclr?)
                                  -- binding parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- bindtyp = envid(see base types above)               binding type -->
<!-- bindclr = envid(see base types above)              binding color -->
<!-- trimnam = envid(see base types above)             named trimming -->
<!ELEMENT trimprm     - -  ( refrsz? & refredg? & jogedge? & trimoff
                           & trimdim )
                                  -- trimming parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- trimoff = dimension(see useful types above)                     -->
<!-- trimdim = xydimensions(see useful types above)                  -->
<!-- diecnam = envid(see base types above)          named die cutting -->
<!ELEMENT diecprm     - -  ( refrsz? & refredg? & jogedge? & diecpos
                           & diecnm )
                                  -- die cutting parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- diecpos = xydimension(see useful types above)                   -->
<!-- diecnm = envid(see base types above)                 die cut name -->
<!-- pnchnam = envid(see base types above)             named punching -->
<!ELEMENT pnchprm     - -  ( refrsz? & refredg? & jogedge? & procoff?
                           & headloc? & pnchdia? )
                                  -- punching parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- procoff = dimension(see useful types above)                     -->
<!-- headloc(see useful types above)                                 -->
<!-- pnchdia = dimension(see useful types above)                     -->
<!-- perfnam = envid(see base types above)          named perforating -->
<!ELEMENT perfprm     - -  ( refrsz? & refredg? & jogedge? & procoff
                           & headloc & perftyp )
                                  -- perforating parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- procoff = dimension(see useful types above)                     -->
<!-- headloc(see useful types above)                                 -->
<!-- perftyp = envid(see base types above)           perforation type -->
<!-- slitnam = envid(see base types above)             named slitting -->
<!ELEMENT slitprm     - -  ( refrsz? & refredg? & jogedge? & headloc )
                                  -- slitting parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- headloc(see useful types above)                                 -->
<!-- nsrtnam = envid(see base types above)            named inserting -->
<!ELEMENT nsrtprm     - -  ( refredg? & jogedge? & nsrtlst )
                                  -- inserting parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- procoff = dimension(see useful types above)                     -->
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<!-- headloc(see useful types above)                                    -->
<!ELEMENT nsrtlst     - -  (nsrtsht)+          -- insert sheet list -->
<!ENTITY % nsrtid   "(nsrtnm | nsrtbin)"       -- insert identifier -->
<!ELEMENT nsrtsht     - -  (%nsrtid; &  nsrtmsg?)
                                  -- insert sheet (any order) -->
<!ATTLIST nsrtsht
                          topsurf  (top | bottom)
                                        #IMPLIED
                            -- insert top surface (enumerated type) --
                          after    NUMBER    #REQUIRED
                                  -- insert after (cardinal) -->
<!-- nsrtnm = envid(see base types above)                 insert name -->
<!ELEMENT nsrtbin     - O  EMPTY >
<!ATTLIST nsrtbin
                          bin      NUMBER    #REQUIRED
                                 -- insert bin (cardinal) -->
<!ELEMENT nsrtmsg     - -  CDATA
                              -- insert message (printable string) -->
<!-- covrnam = envid(see base types above)              named covers -->
<!ELEMENT covrprm     - -  ( refrsz? & refredg? & jogedge? & frntcvr?
                          & backcvr?)
                                -- covers parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!ELEMENT frntcvr     - -  (covrnm | medspc)          -- frontcover -->
<!ELEMENT backcvr     - -  (covrnm | medspc)          -- back cover -->
<!-- covrnm = envid(see base types above)                cover name -->
<!-- foldnam = envid(see base types above)               med folding -->
<!ELEMENT foldprm     - -  ( refrsz? & refredg? & jogedge? & headloc? )
                               -- folding parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- headloc(see useful types above)                                 -->
<!ELEMENT ofinspc     - -  (finopnm & finoprm?)
                            -- other finishing specification (any order) -->
<!-- finopnm = pubobid(see base types above)       finishing op name -->
<!ELEMENT finoprm     - -  (nfinprm?, sfinprm?)
                                       -- finishing op parameters -->
<!-- nfinprm = envid(see types above)  named finishing op parameters -->
<!ELEMENT sfinprm     - -  ( refrsz? & refredg? & jogedge? & procoff?
                          & headloc? & addlprm? )
                                -- stitching parameters (any order) -->
<!-- refrsz = xydimension(see useful types above)                    -->
<!-- procoff = dimension(see useful types above)                     -->
<!-- headloc(see useful types above)                                 -->
<!ELEMENT addlprm     - -  ANY
            -- additional parameters (depends on finishing op name) -->
<!-- Management Document Production Instructions                    -->
<!ENTITY % colrspc  "(colrsid | colrdsc)"
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                                        -- colorant set specification -->
<!ELEMENT colrdpi      - -  (colrmsg?, %colrspc;)
                                              -- colorant set dpi -->
<!ELEMENT colrmsg      - -  CDATA
                           -- colorant set message (printable string) -->
<!-- colrsid = envid(see base types above)  colorant set identifier -->
<!ELEMENT colrdsc      - -  (colrcls, colrid+)
                                         -- colorant set description -->
<!-- colrcls = pubobid(see above)     colorant set class identifier -->
<!-- colrid = envid(see base types above)       colorant identifier -->
<!ELEMENT dstrdpi      - -  CDATA
                    -- document start message dpi (printable string) -->
<!ELEMENT auxpdpi      - -  (strpage? & seppage? & endpage?)
                                         -- auxilliary page type dpi -->
<!-- strpage = envid(see base types above)          start page type -->
<!-- seppage = envid(see base types above)      separator page type -->
<!-- endpage = envid(see base types above)            end page type -->
<!ELEMENT dcmtdpi      - -  CDATA
                           -- document comment dpi (printable string) -->
<!ELEMENT timedpi      - O  EMPTY>
<!ATTLIST timedpi
                           timeout  NUMBER    #REQUIRED
                                       -- timeout dpi (cardinal) -->
<!-- abrtdpi = envid(see base types above)         abort policy dpi -->
<!ELEMENT outbdpi      - -  ((outbnam | outbnum)?) -- output bin dpi -->
<!ATTLIST outbdpi
                           outposn  NUMBER    #IMPLIED
                                -- output positions (cardinal) -->
<!-- outbnam = envid(see base types above)          output bin name -->
<!ELEMENT outbnum      - O  EMPTY>
<!ATTLIST outbnum
                           binnum   NUMBER    #REQUIRED
                                  -- output bin number (cardinal) -->
<!ELEMENT ospcdpi      - -  (addlout*)   -- output specification dpi -->
<!ATTLIST ospcdpi
                          collate  %Boolean; #IMPLIED
                                           -- collated (boolean) --
     -- he only allowed values are "true" or "false", in lowercase--
                           offset   %Boolean; #IMPLIED
                                             -- offset (boolean) --
    -- The only allowed values are "true" or "false", in lowercase--
                           burst    %Boolean; #IMPLIED
                                              -- burst (boolean) --
    -- The only allowed values are "true" or "false", in lowercase-->
<!ELEMENT addlout      - -  (ospcnm,ospcvl?)    -- additional output -->
<!-- ospcnm = pubobid(see types above)    output specification name -->
<!ELEMENT ospcvl       - -  ANY         -- output specification value -->
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642830 21T


```
<!--output specification value depends on output specification name -->
<!ELEMENT denddpi      - -  CDATA
                      -- document end message dpi (printable string) -->
```

37.2 Clear Text Content Representation and Interchange Format

The SPDL content consists of sequences of tokens. Each token represents a unit of information which is processed by the content processor. This clause specifies the clear text encoding of the tokens in SPDL content.

37.2.1 Token structure and delimiters

Each token is encoded by a sequence of one or more ISO 646-encoded octets, hereafter referred to as characters. The sequence of characters encoding each token is terminated by some delimiter. The white space characters Null, Tab, Line Feed, Form Feed, Carriage Return, or Space are delimiters except when they occur within Strings, *DataBlocks* and, in the case of space, within Comments. The encoding for the token that follows a white space delimiter begins with the first following character that is not a white space character. The characters (,), <, >, [,], {, }, /, and % are special The characters (, <, and / delimit the encoding of the preceding token and begin the encoding of a new token. The characters [,], {, }, and the character sequences < < and > > are self delimiting tokens; they terminate the encoding of any preceding token, they are the encoding of a token themselves, and the encoding of the following token begins with the first following character that is not a whitespace character. The character) terminates the encoding of a token begun with the character (and the encoding of the following token begins with the first following character that is not a white space character. Except when it occurs in the self delimiting token > >, the character > terminates the encoding of a token begun with the character < and the encoding of the following token begins with the first following character that is not a white space character.

Any occurrence of the character % outside an OctetString, DataBlock, or DataBlockContinuation token introduces a Comment. The comment consists of all characters between the % and the next Carriage Return, Line Feed, or Form Feed character. The % character and the comment it delimits are together treated as one white space character.

37.2.2 Token types

There are two types of tokens in the clear text encoding of content: *base tokens* and *composite tokens*.

Base tokens do not contain other tokens. The processing of the characters in the SPDL content produces base tokens Examples of base tokens include those for *OctetStrings*, Integers, and Names.

A composite token is a token constructed from a sequence of base and/or other composite tokens. When a composite token is encountered in content, the parser processes each of the contained tokens as it is encountered, then processes the containing composite token The only composite tokens in the clear text encoding of content are those for Procedures.

37.2.3 Encodings for content object types

The following subclauses detail the clear text encoding for each of the SPDL object types. The encoding for objects of type *Operator* is described in annex A.

37.2.3.1 Boolean

There is no representation for the type *Boolean* in the Clear Text Content Representation and Interchange Format. Objects of type *Boolean* may be created in content via execution of the operators **True** and **False**.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642831 156

37.2.3.2 Dictionary

There is no representation for the type *Dictionary* in the Clear Text Content Representation and Interchange Format. Objects of type *Dictionary* may be created in content via execution of the **MakeDictionary** or **MakeandStoreDictionary** operators.

The self-delimiting operator tokens << and >> (representing **Mark** and **MakeandStoreDictionary**, respectively; see annex A) may be used to clearly demark the creation of a Dictionary in content:

<<token token ... token >>

37.2.3.3 Integer

There are two representations for Integers in the Clear Text Content Representation and Interchange Format. One representation consists of an optional sign followed immediately by one or more decimal digits. The number is interpreted as a signed decimal integer and converted to an SPDL Integer object.

An alternate representation takes the form *base#number* where *base* is a decimal integer between 2 and 36, and *number* is then interpreted in that base; it must consist of digits in the range 0 to *base*-1, inclusive. Digits greater than 9 are represented by the characters A through Z (or a through z). The number is treated as an unsigned integer in the given base and is converted to an SPDL Integer object.

If the number represented in either representation exceeds the implementation-dependent limit for the size of an Integer, it is converted to an SPDL Real object before being pushed on the operand stack.

37.2.3.4 Mark

There is no representation for the type *Mark* in the Clear Text Content Representation and Interchange Format. Objects of type *Mark* may be created in content via execution of the operator **Mark**.

37.2.3.5 Name

There are two representations for objects of type *Name*: one for executable Names and one for literal Names..

37.2.3.5.1 Executable Name

The representation for an executable Name consists of a character from the latin alphabet: "a", "b", "c", . . . , "z"; "A", "B", "C", . . . , "Z"; or a period ".". Each succeeding character in the sequence may be chosen from the latin alphabet letters, decimal digits "0", "1", "2", . . . , "9", the underbar "_", the ":", or the period ".".

37.2.3.5.2 Literal Name

The representation for a literal Name consists of the character /, followed immediately by a character from the latin alphabet: "a", "b", "c", . . . , "z"; "A", "B", "C", . . "Z"; or a period ".". Each succeeding character in the sequence may be chosen from the latin alphabet letters, decimal digits "0", "1", "2", . . . , "9", the underbar "_", the ":", or the period ".".

37.2.3.6 Null

There is no representation for the type *Null* in the Clear Text Content Representation and Interchange Format. Objects of type *Null* may be created in content via execution of the operator **Null**.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642832 092

37.2.3.7 OctetString

There are three representations for an OctetString: a Character String, a Hexadecimal encoded String and an ASCII85 encoded String.

37.2.3.7.1 Character String

An *OctetString* may be represented in the form

(*xyz*)

where *xyz* represents a sequence of characters that represent zero or more octets. Unless preceded by the character \, each character between the parentheses represents the octet which is the ISO 646 coding of that character. The OctetString shall be delimited solely by the surrounding parentheses; white space characters occuring in the OctetString are treated as characters, not as delimiters. Balanced parentheses () with zero or more included octets may appear in the OctetString token; both parentheses and the characters between them are treated as characters and not as a embedded OctetString.

The \ character is an escape character that may be used to quote unbalanced parentheses, non-printing characters, and the \ character itself. The character immediately following the \ determines its interpretation:

\r Carriage Return

\t Horizontal Tab

\b *BackSpace*

\f Form Feed

\\ \

\((

\))

ddd *Arbitrary Octet*

If the character following the \ is not in the preceding list, the \ is ignored and the character following the \ is interpreted normally. In the *ddd* form, each of the *d*'s shall be an octal digit and the first *d* shall be 3 or less. The *ddd* form is interpreted as the single octet whose value in octal is *ddd* This form may be used to include an arbitrary octet in the OctetString.

37.2.3.7.2 Hexadecimal encoded String

An OctetString may be represented in the form

<*xyz*>

where *xyz* represents a sequence of zero or more hexadecimal digits, the characters 0-9 and A-F or a-f, each pair of which represents one octet in the OctetString. If there is an odd number of hexadecimal digits, a final 0 digit is assumed as the second hexidecimal digit of the last octet. While space characters occuring in the Hexadecimal encoded String are ignored and are not treated as delimiters. If the Hexadecimal encoded String is malformed, then **RaiseError** is invoked with *SyntaxError* as its operand.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642833 T29

37.2.3.7.3 ASCII85 encoded String

An OctetString may be represented in the form

< *xyz* >

where *xyz* represents a sequence of zero or more ASCII85 encoded octets (see 31.2.2) in which the sub-sequence "<~", the ISO 8879 ETAGO in the Reference Syntax, shall not occur. If an ASCII85 encoding of an *OctetString* would have occurances of the sub-sequence "<~", these shall be eliminated by inserting a blank character between the "<" and the "~". If the ASCII85 encoded String is malformed, then **RaiseError** is invoked with *SyntaxError* as its operand.

37.2.3.8 Path

There is no representation for Paths in the Clear Text Content Representation and Interchange Format. Paths may be created in content via execution of the path construction operators described in clause 29.

37.2.3.9 Real

The representation of a Real in the Clear Text Content Representation and Interchange Format takes the form of an optional sign, followed immediately by a sequence of one or more decimal digits with an embedded or leading full stop (decimal point), a trailing exponent, or both. The exponent, if present, consists of the character E or e, followed immediately by an optional sign, followed immediately by one or more decimal digits. This decimal value is converted to a binary value as specified in IEEE 754-1987. If the Real is outside the range of Reals representable within the imiplementation, then **RaiseError** is invoked with *LimitCheck* as its operand.

37.2.3.10 SaveObjects

There is no representation for SaveObjects in the Clear Text Content Representation and Interchange Format. SaveObjects may be created in content via execution of the **SaveState** operator.

37.2.3.11 StreamObject

There is no representation for objects of the type *StreamObject* in the Clear Text Content Representation and Interchange Format. Objects of type *StreamObject* may be created in content via execution of the **Filter** operator or the **FindResource** operator.

37.2.3.12 Vector

There is a representation for objects of type *Vector* which have an executable object reference; such object references are of type *Procedure*. There is no representation for objects of type *Vector* which have a literal object reference.

37.2.3.12.1 Vector with a literal object reference

Objects of type *Vector* which have a literal object reference may be created in content via execution of the **MakeVector** or **MakeandStoreVector** operators.

The self-delimiting operator tokens [and] (representing **Mark** and **MakeandStoreVector**, respectively; see annex A) may be used to clearly demark the creation of a literal Vector in content:

[token token ... token]

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642834 965

37.2.3.12.2 Procedure (executable VectorReference)

The Procedure token is a composite token. The representation for a Procedure, or executable *VectorReference*, of length N in the Clear Text Content Representation and Interchange Format takes the form

{token1 token2 ... tokenN}

where token I is decoded to produce a simple value or object reference that becomes element I of the executable Vector. Token execution is suspended during the processing of the whole composite token. Each token, token1 ... tokenN, may be either a base token or a composite token. Composite tokens result from processing Procedure tokens that are embedded in the sequence of characters between the matching { and } delimiters. Procedure tokens can be nested within Procedure tokens to arbitrary depth.

37.2.3.13 Operators

With the exception of the self-delimiting tokens [,], <<, and >>, there is no representation for objects of type *Operator* in the clear text encoding. Operator objects are bound to Names in *SystemDict*, and may be accessed in content via their Names. annex A lists each SPDL operator and the Name (and, if applicable, the self-delimiting token) by which it may be accessed in the clear text encoding.

37.2.4 In-line Raster Graphic Image Data Tokens

There is a token type used to encode raster graphic image Data Sources inline in clear text documents. The use of this token type is discussed in clause 28 .

37.2.4.1 DataBlock Token

A *DataBlock* token is represented in the form

<|xyz|>

where *xyz* represents a sequence of zero or more ASCII85-encoded octets (see 31.2.2) in which the sub-sequence "<|", the ISO 8879 ETAGO in the Reference Syntax, shall not occur. If an ASCII85 encoding of a Data Source would have occurances of the sub-sequence "<|", these shall be eliminated by inserting a blank character between the "<" and the "|". If the ASCII85 encoded String is malformed, then **RaiseError** is invoked with *SyntaxError* as its operand. There is no limit on the length of a *DataBlock*.

38 Binary Representation and Interchange Format

The Binary Representation and Interchange Format consists of the Binary Structure Representation and Interchange Format and the Binary Content Representation and Interchange Format.

38.1 Binary Structure Representation and Interchange Format

The Binary Structure Representation and Interchange Format is based on ASN.1. The ASN.1 types which are used to represent base structure element values are described in 38.1.1.

The ASN.1 modules which comprise the structure definition are specified in 38.2 through 38.2.6. The encoding of the SPDL structure shall be as defined in ISO/IEC 8825. All tags which are optional by virtue of IMPLICIT declarations shall be omitted in the Binary Structure Representation and Interchange Format.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642835 8T1

38.2 Base Structure Element Value Types

The encoding of the base structure element value types shall be as follows.

38.2.1 Boolean type

Values of the *Boolean* type are encoded by values of the BOOLEAN type.

38.2.2 Enumerated types

Values of the *Enumerated* types are encoded by values of specific ENUMERATED types.

38.2.3 Integer type

Values of the *Integer* type are encoded by values of the INTEGER type.

38.2.4 Real type

Values of the *Real* type are encoded by values of the REAL type.

38.2.5 Printable String type

Values of the *Printable* String type are encoded by values of the PrintableString type.

38.2.6 Octet String type

Values of the *Octet String* type are encoded by values of the OctetString type.

38.2.7 Name type

Values of the *Name* type are encoded by values of the VisibleString type.

38.2.8 Object Identifier type

Values of the *Object Identifier* type are encoded by values of the OBJECT IDENTIFIER type.

38.2.9 Public Identifier type

Values of the *Public Identifier* type are encoded by values of the VisibleString type.

38.2.10 Environment Name type

Values of the *Environment Name* type are encoded by values of the VisibleString type.

38.2.11 Structured Name type

Values of the *Structured Name* type are encoded by instances of the Structured-Name type as defined by ISO/IEC 9541-2 annex A.

38.2.12 Comments

Comments are encoded by values of the IA5String type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642836 738

38.3 SPDL Structure Definition Module

```
ISO10180-SPDL  {iso(1) standard(0) 10180 modules(1) spdl(0)}
DEFINITIONS                    ::= BEGIN
EXPORTS                        Integer, Cardinal, Positive-Integer, Real,
                               Number, Non-Negative-Number, Printable-String, Name,
                               Environment-ID, Public-Object-ID, Public-Identifier,
                               Environment-Name, Comment, SPDL-Instance,
                               Top-Level-Structure,TokenSequence;
IMPORTS                        Document, Pageset, Picture
                               FROM ISO10180-SPDL-DOC
                               {iso(1) standard(0) 10180 modules(1) document(1)}
                               Includable-Structure FROM ISO10180-SPDL-EXTREF
                               {iso(1) standard(0) 10180 modules(1) extref(2)}
                               Environment-Resource FROM ISO10180-SPDL-RESOURCE
                               {iso(1) standard(0) 10180 modules(1) resource(3)}
                               spdl-external-id FROM ISO10180-SPDL-OID
                               {iso(1) standard(0) 10180  modules(1) oid(6)} ;
-- Base types
ub-integer                     INTEGER          ::=     2147483647  -- 2^31-1 --
Integer                        ::= INTEGER (-(ub-integer)..ub-integer)
Cardinal                       ::= Integer (0..MAX)
Positive-Integer               ::= Integer (1..MAX)
Real                           ::= REAL
-- Includes all basic single width floating point --
-- binary numbers as specified by IEE 754 --
Number                         ::= CHOICE {
     integer                           Integer,
     real                              Real }
Non-Negative-Number            ::= CHOICE {
     integer-valued                    Cardinal,
     real-valued                       Real (0<..MAX) }
Printable-String               ::= PrintableString(SIZE(127))
Name                           ::= [APPLICATION 1] IMPLICIT VisibleString
                                   (FROM ( "A".."Z" | "a".."z" |"0".."9" |
                                   "_" | "-" | ":" | "." ))
                                   -- Beginning with "A".."Z","a".."z", or "." --
Public-Identifier              ::= [APPLICATION 2] IMPLICIT VisibleString (SIZE(218))
                                   (FROM ("A".."Z" | "a".."z" | "0".."9" |
                                   "'" | "(" | ")" | "+" | "," |
                                   "-" | " " | ":" | "=" | "?" |
                                   "/" ))
                                   -- Satisfying the public identifier syntax --
                                   -- defined by ISO/IEC 9070 --
Public-Object-ID               ::= CHOICE {
     public-identifier                 Public-Identifier,
     object-identifier                 OBJECT IDENTIFIER }
Environment-Name               ::= [APPLICATION 3] IMPLICIT VisibleString (SIZE(100))
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642837 674


```
                                (FROM ( "A".."Z"| "a".."z" | "0".."9" |
                                "'" | "(" | ")" | "+" | "," |
                                "-" | " " | ":" | "=" | "?" |
                                "/" ))
                                -- Satisfying the object-name syntax --
                                -- defined by ISO/IEC 9070 --
Environment-ID                  ::= CHOICE {
     environment-name                   Environment-Name,
     public-object-id                   Public-Object-ID }
Comment                         ::= [APPLICATION 0] IMPLICIT IA5String (SIZE(127))
--  SPDL Instance
SPDL-Instance                   ::= EXTERNAL (WITH COMPONENTS {
     direct-reference                   (spdl-external-id) PRESENT,
     indirect-reference                 ABSENT,
     data-value-descriptor              ABSENT,
     encoding                           (WITH COMPONENTS  {
        single-ASN1-type                   (INCLUDES Top-Level-Structure),
        octet-aligned                      ABSENT,
        arbitrary                          ABSENT }) PRESENT })
--  Top Level Structure
Top-Level-Structure             ::= CHOICE {
     document                           Document,
     resource                           Environment-Resource,
     includable                         Includable-Structure }
--  Block
Block                           ::= CHOICE {
     pageset                            Pageset,
     picture                            Picture,
     resource                           Environment-Resource }
--  TokenSequence
TokenSequence                   ::= [APPLICATION 4] IMPLICIT OCTET STRING
END
```

38.4 SPDL Document Module

```
ISO10180-SPDL-DOC  {iso(1)standard(0) 10180 modules(1) document(1)}
DEFINITIONS                     ::= BEGIN
EXPORTS                         Document, Pageset, Picture, Picture-Body,
                                Non-SPDL-Picture-Body, Prologue, Informative-Dec, Hint,
                                Context-Declaration, Context-Addition, Setup-Proc;
IMPORTS                         Environment-ID, Comment, TokenSequence
                                FROM ISO10180-SPDL
                                {iso(1) standard(0) 10180 modules(1) spdl(0)}
                                External-Declaration, External-Reference
                                FROM ISO10180-SPDL-EXTREF
                                {iso(1) standard(0) 10180  modules(1) extref(2)}
                                Resource-Def, Resource-Declaration, Internal-Resource-ID
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642838 500


```
                                FROM ISO10180-SPDL-RESOURCE
                                {iso(1) standard(0) 10180  modules(1) resource(3)}
                                DPI-Declaration
                                FROM ISO10180-SPDL-DPI
                                {iso(1) standard(0) 10180  modules(1) dpi(5)}
                                content-spdl-binary
                                FROM ISO10180-SPDL-OID
                                {iso(1) standard(0) 10180  modules(1) oid(6)};
-- High Level Structure Elements
Document                        ::= CHOICE {
     pageset                            Pageset,
     picture                            Picture }
Pageset                         ::= [APPLICATION 5] IMPLICIT SEQUENCE {
     comment                           Comment OPTIONAL,
     prologue-or-ref                [0] CHOICE {
          prologue                          Prologue
          reference                         External-Reference } OPTIONAL,
     body                           [1] IMPLICIT SEQUENCE OF CHOICE {
          pageset                           Pageset,
          picture                           Picture,
          reference                         External-Reference } }
Picture                         ::= [APPLICATION 6] IMPLICIT SEQUENCE {
     comment                            Comment OPTIONAL,
     content-rep-id                     Content-Rep-ID,
     body-or-reference                  CHOICE {
        picture-body                        Picture-Body,
        non-spdl-body                       Non-SPDL-Picture-Body,
        reference                           External-Reference } }
Content-Rep-ID                  ::= OBJECT IDENTIFIER
Picture-Body                    ::= [APPLICATION 7] IMPLICIT SEQUENCE {
     comment                           Comment OPTIONAL,
     prologue-or-ref                [0] CHOICE {
          prologue                          Prologue
          reference                         External-Reference } OPTIONAL,
     body                           [1] IMPLICIT SEQUENCE OF CHOICE {
          picture                           Picture,
          tokensequence                     TokenSequence
          reference                         External-Reference } }
Non-SPDL-Picture-Body           ::= [APPLICATION 33] ANY
-- Prologue Structure Elements
Prologue                        ::= [APPLICATION 8] IMPLICIT SEQUENCE {
     comment                           Comment OPTIONAL,
     external-dec                   [0] IMPLICIT SEQUENCE OF External-Declaration
                                        OPTIONAL,
     informative-dec-or-ref         [1] CHOICE {
        informative-dec                     Informative-Dec,
        reference                           External-Reference } OPTIONAL,
     non-spdl-operation             [2] IMPLICIT SEQUENCE OF Non-Spdl-Operation
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642839 447


```
                                    OPTIONAL,
   dpi-dec-or-ref                [3] CHOICE {
      dpi-declaration                    DPI-Declaration,
      reference                          External-Reference } OPTIONAL,
   context-dec-or-ref            [4] CHOICE {
      context-declaration                Context-Declaration,
      reference                          External-Reference } OPTIONAL,
   resource-def-or-dec           [5] IMPLICIT SEQUENCE OF CHOICE {
      resource-def                       Resource-Def,
      resource-dec                       Resource-Declaration,
      context-add                        Context-Addition,
      reference                          External-Reference } OPTIONAL,
   setup-proc-or-ref             [6] CHOICE {
      setup-proc                         Setup-Proc
      reference                          External-Reference } OPTIONAL }
   Informative-Dec               ::= [APPLICATION 9] IMPLICIT SEQUENCE {
      comment                          Comment OPTIONAL,
      set-of-hints                  [0] IMPLICIT SET OF CHOICE {
           hint                          Hint,
           reference                     External-Reference } }
   Hint                          ::= [APPLICATION 10] IMPLICIT SEQUENCE {
      comment                          Comment OPTIONAL,
      hint-name                     [0] Environment-ID,
      hint-value                    [1] ANY }
   Non-Spdl-Operation            ::= SEQUENCE {
        comment                        Comment OPTIONAL,
        operation-name              [0] Environment-ID,
        operation-value             [1] ANY }
   Context-Declaration           ::= [APPLICATION 11] IMPLICIT SEQUENCE {
        comment                        Comment OPTIONAL,
        dictionary-identifiers         SEQUENCE OF Internal-Resource-ID }
   Context-Addition              ::= [APPLICATION 12] IMPLICIT SEQUENCE {
        comment                        Comment OPTIONAL,
        dictionary-identifier          Internal-Resource-ID }
   Setup-Proc                    ::= [APPLICATION 13] IMPLICIT SEQUENCE {
        comment                        Comment OPTIONAL,
        body                           SEQUENCE OF CHOICE {
             tokensequence                 TokenSequence
             reference                     External-Reference } }
END
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

38.5 External Reference Module

```
ISO10180-SPDL-EXTREF  {iso(1) standard(0) 10180 modules(1) extref(2)}
DEFINITIONS                    ::= BEGIN
EXPORTS                        Includable-Structure,External-Declaration,
                               External-Reference, Location-Identifier;
IMPORTS                        Name, Public-Object-ID, TokenSequence
                               FROM ISO10180-SPDL
                               {iso(1) standard(0) 10180  modules(1) spdl(0)}
                               Pageset, Picture, Picture-Body, Non-SPDL-Picture-Body,
                               Prologue, Informative-Dec, Hint, Context-Declaration,
                               Setup-Proc FROM ISO10180-SPDL-DOC
                               {iso(1) standard(0) 10180  modules(1) document(1)}
                               Resource-Def, Dictionary-Spec, Data-Source-Spec,
                               Color-Space-Spec, Pattern-Spec, Form-Spec
                               FROM ISO10180-SPDL-RESOURCE
                               {iso(1) standard(0) 10180  modules(1) resource(3)}
                               Font-Spec, Font-Ref-Spec, Font-Reference, Font-Properties
                               Font-Index-Map, Fonttype1-Font-Spec, Remapped-Font-Spec,
                               Glyph-Index-Table, Composite-Font-Spec, User-Font-Spec,
                               Glyph-Index-Map-Spec
                               FROM ISO10180-SPDL-FONTS
                               {iso(1) standard(0) 10180  modules(1) fonts(4)}
                               DPI-Declaration, Additional-DPI
                               FROM ISO10180-SPDL-DPI
                               {iso(1) standard(0) 10180  modules(1) dpi(5)}
                               Font-Attribute-Set
                               FROM ISO-STANDARD-9541-FONT-ATTRIBUTE-SET
                               {iso(1) standard(0) 9541 2 2}
                               Dor FROM DOR-definition
                               {joint-iso-ccitt(2) dor(11) reference-definition(0)};
-- Includable Structure
Includable-Structure           ::= CHOICE {
     pageset                           Pageset,
     picture                           Picture,
     picture-body                      Picture-Body,
     non-spdl-pict                     Non-SPDL-Picture-Body,
     prologue                          Prologue,
     informative-dec                   Informative-Dec,
     hint                              Hint,
     context-dec                       Context-Declaration,
     resource-def                      Resource-Def,
     setup-proc                        Setup-Proc,
     tokensequence                     TokenSequence,
     font-reference                    Font-Reference,
     dictionary-spec                   Dictionary-Spec,
     font-spec                         Font-Spec,
     font-ref-spec                     Font-Ref-Spec,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
      fonttype1-spec                  Fonttype1-Font-Spec,
      remapped-font-spec              Remapped-Font-Spec,
      composite-font-spec             Composite-Font-Spec,
      user-font-spec                  User-Font-Spec,
      glyph-index-map-spec            Glyph-Index-Map-Spec,
      glyph-index-table               Glyph-Index-Table,
      font-index-map                  Font-Index-Map,
      data-source-spec                Data-Source-Spec,
      color-space-spec                Color-Space-Spec,
      pattern-spec                    Pattern-Spec,
      form-spec                       Form-Spec,
      dpi-declaration                 DPI-Declaration,
      additional-dpi                  Additional-DPI }
-- External Declaration
External-Declaration            ::= SEQUENCE {
      comment                         Comment OPTIONAL,
      structure-class-id              Structure-Class-ID,
      structure-id                    External-Reference,
      structure-location              Location-Identifier }
Structure-Class-ID              ::= ENUMERATED {
                                    pageset (0),
                                    picture (1),
                                    picture-body (2),
                                    non-spdl-picture-body (3),
                                    prologue (4),
                                    informative-declaration (5),
                                    hint (6),
                                    context-declaration (7),
                                    resource-definition (8),
                                    dictionary-spec (9),
                                    setup-procedure (10),
                                    tokensequence (11),
                                    font-reference (12),
                                    font-attribute-set (13),
                                    font-spec (14),
                                    font-ref-font-spec (15),
                                    fonttype1-font-spec (16),
                                    remapped-font-spec (17),
                                    composite-font-spec (18),
                                    user-font-spec (19),
                                    glyph-index-map-spec (20),
                                    gitt (21),
                                    font-index-map (22),
                                    color-space-spec (23),
                                    data-source-spec (24),
                                    pattern-spec (25),
                                    form-spec (26),
                                    dpi-declaration (27),
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

```
                                        additional-dpi (28) }
-- Location Identifier
Location-Identifier             ::= CHOICE {
     public-id                      [0] Public-Object-ID,
     local-location-id              [1] OCTET STRING,
     encoding-dependent                 CHOICE {
        dor-identifier                  [2] Dor,
        identified-syntax               [3] Identified-Syntax-Loc } }
Identified-Syntax-Loc           ::= EXTERNAL (WITH COMPONENTS {
     direct-reference                   PRESENT,
     indirect-reference                 ABSENT,
     data-value-descriptor              ABSENT,
     encoding                           (WITH COMPONENTS {
        single-ASN1-type                     PRESENT
        octet-aligned                        ABSENT,
        arbitrary                            ABSENT } ) PRESENT } )
END
```

38.6 Resource Definition Module

```
ISO10180-SPDL-RESOURCE  {iso(1) standard(0) 10180  modules(1) resource(3)}
DEFINITIONS                     ::= BEGIN
EXPORTS                         External-Resource-ID, Internal-Resource-ID,
                                Environment-Resource, Resource-Def,
                                Resource-Declaration, Dictionary-Spec, Data-Source-Spec,
                                Color-Space-Spec, Pattern-Spec, Form-Spec;
IMPORTS                         Name, Environment-ID, Public-Object-ID, Comment,
                                TokenSequence
                                FROM ISO10180-SPDL
                                {iso(1) standard(0) 10180  modules(1) spdl(0)}
                                Informative-Dec, Context-Addition,
                                Setup-Proc FROM ISO10180-SPDL-DOC
                                {iso(1) standard(0) 10180  modules(1) document(1)}
                                External-Declaration, External-Reference,
                                Location-Identifier
                                FROM ISO10180-SPDL-EXTREF
                                {iso(1) standard(0) 10180  modules(1) extref(2)}
                                Font-Spec, Glyph-Index-Map-Spec
                                FROM ISO10180-SPDL-FONTS
                                {iso(1) standard(0) 10180  modules(1) fonts(4)};
 -- Useful Types
External-Resource-ID            ::= Environment-ID
Internal-Resource-ID            ::= Name
-- Environment Resource
Environment-Resource            ::= [APPLICATION 15] IMPLICIT SEQUENCE {
     comment                            Comment OPTIONAL,
     external-dec                   [0] IMPLICIT SEQUENCE OF External-Declaration
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642843 978


```
                                        OPTIONAL,
      informative-dec-or-ref        [1] CHOICE {
           informative-dec                  Informative-Dec,
           reference                        External-Reference } OPTIONAL,
      resource-or-context           [2] IMPLICIT SEQUENCE OF CHOICE {
           resource-dec                     Resource-Declaration,
           context-add                      Context-Addition,
           reference                        External-Reference } OPTIONAL,
      resource-def-or-undef         [3] CHOICE {
           resource-def                     Resource-Def,
           resource-undef                   Resource-Undef,
           reference                        External-Reference } }
Resource-Undef                  ::= SEQUENCE {
      comment                          Comment OPTIONAL,
      resource-class-id             [0] IMPLICIT Resource-Class-ID,
      external-resource-id          [1] External-Resource-ID }
Resource-Def                    ::= [APPLICATION 16] IMPLICIT SEQUENCE {
      comment                          Comment OPTIONAL,
      resource-class-id             [0] Resource-Class-ID,
      external-resource-id          [1] External-Resource-ID,
      resource-spec-or-ref          [2] CHOICE {
         resource-spec                    Resource-Spec,
         reference                        External-Reference } }
Resource-Class-ID               ::= ENUMERATED {
                                        dictionary (1),
                                        indexed-font (2),
                                        glyph-index-map (3),
                                        color-space (4),
                                        data-source (5),
                                        filter (6),
                                        pattern (7),
                                        form (8) }
-- Resource Declaration
Resource-Declaration            ::= SEQUENCE {
      comment                          Comment OPTIONAL,
      resource-class-id             [0] IMPLICIT
      internal-resource-id          [1] IMPLICIT
      external-resource-id          [2] External-Resource-ID }
-- Resource Specifications
Resource-Spec                       CHOICE {
      dictionary-spec                   Dictionary-Spec,
      font-spec                         Font-Spec,
      glyph-map-spec                    Glyph-Index-Map-Spec,
      color-space-spec                  Color-Space-Spec,
      data-source-spec                  Data-Source-Spec,
      pattern-spec                      Pattern-Spec,
      form-spec                         Form-Spec }
Dictionary-Spec                 ::= [APPLICATION 17] IMPLICIT SEQUENCE {
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642844 804


```
      comment                          Comment OPTIONAL,
      specification                    SEQUENCE OF CHOICE {
         tokensequence                     TokenSequence
         reference                         External-Reference } }
Color-Space-Spec                  ::= [APPLICATION 18] IMPLICIT SEQUENCE {
      comment                          Comment OPTIONAL,
      color-space-family             [0] Public-Object-ID,
      primary-set-spec                 CHOICE {
         primary-set-id                [1] Public-Object-ID
         primary-list                  [2] IMPLICIT SEQUENCE OF Public-Object-ID }
                                       OPTIONAL,
      specification                  [3] IMPLICIT SEQUENCE OF CHOICE {
           tokensequence                 TokenSequence
           reference                     External-Reference } }
Data-Source-Spec                  ::= [APPLICATION 19] IMPLICIT SEQUENCE {
      comment                          Comment OPTIONAL,
      specification                    CHOICE {
         location-identifier           [0] Location-Identifier,
         data-block                    [1] OCTET STRING } }
Pattern-Spec                      ::= [APPLICATION 20] IMPLICIT SEQUENCE {
      comment                          Comment OPTIONAL,
      specification                    SEQUENCE OF CHOICE {
         tokensequence                      TokenSequence
         reference                          External-Reference } }
Form-Spec                         ::= [APPLICATION 21] IMPLICIT SEQUENCE {
      comment                          Comment OPTIONAL,
      specification                    SEQUENCE OF CHOICE {
         tokensequence                      TokenSequence
         reference                          External-Reference } }
END
```

38.7 Fonts and Glyph Index Map Module

```
ISO10180-SPDL-FONTS  {iso(1) standard(0) 10180  modules(1) fonts(4)}
DEFINITIONS          ::= BEGIN
EXPORTS              Glyph-ID, Font-Spec, Font-ID-Spec,
                     Font-Ref-Spec, Font-Reference,
                     Fonttype1-Font-Spec, Remapped-Font-Spec,
                     Glyph-Index-Table, Composite-Font-Spec,
                     Font-Index-Map-Spec,  User-Font-Spec,
                     Glyph-Index-Map-Spec;
IMPORTS              Public-Object-ID, Comment
                     FROM ISO10180-SPDL
                     {iso(1) standard(0) 10180  modules(1) spdl(0)}
                     External-Reference
                     FROM ISO10180-SPDL-EXTREF
                     {iso(1) standard(0) 10180  modules(1) extref(2)}
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642845 740


```
                    External-Resource-ID
                    FROM ISO10180-SPDL-RESOURCE
                    {iso(1) standard(0) 10180  modules(1) resource(3)}
                    Structured-Name
                    FROM ISO9541-SN
                    {iso(1) standard(0) 9541 2 3}
                    Font-Attribute-Set
                    FROM ISO-STANDARD-9541-FONT-ATTRIBUTE-SET
                    {iso(1) standard(0) 9541 2 2}
--  Glyph Identifiers
Structured-Glyph-Name          ::= Structured-Name ( WITH COMPONENTS {
     owner-name                        PRESENT,
     owner-description                 OPTIONAL,
     object-name                       PRESENT,
     object-description                OPTIONAL } )
Simple-Glyph-Name              ::= Name (SIZE(1..100))
                                   (FROM ( "A".."Z " | "a".."z" | "0".."9" |
                                   "_" | "-" | ":" | "." ))
                                   -- Beginning with "A".."Z " or "a".."z" --
Glyph-ID                       ::= CHOICE {
     structured-name                   [0] Structured-Glyph-Name,
     simple-name                       [1] Simple-Glyph-Name }
--  Font Specifications
Font-Spec                      ::= CHOICE {
     font-id-spec                      Font-ID-Spec,
     font-ref-spec                     Font-Ref-Spec,
     fonttype1-spec                    Fonttype1-Spec,
     remapped-font-spec                Remapped-Font-Spec,
     composite-font-spec               Composite-Font-Spec,
     user-font-spec                    User-Font-Spec }
--  Font Identifier Font Specification
Font-ID-Spec                   ::= [APPLICATION 22] IMPLICIT SEQUENCE {
     comment                           Comment OPTIONAL,
     indexed-font-id                   External-Resource-ID }
--  Font Reference Font Specification
Font-Ref-Spec                  ::= [APPLICATION 23] IMPLICIT SEQUENCE {
     comment                           Comment OPTIONAL,
     font-reference-or-ref        [0]  CHOICE {
        font-reference                     Font-Reference,
        reference                          External-Reference },
     glyph-index-map-id           [1]  Glyph-Index-Map-ID }
--  Font Reference
Font-Reference                 ::= [APPLICATION 24] IMPLICIT SEQUENCE {
     comment                           Comment OPTIONAL,
     font-resource-id             [0]  External-Resource-ID OPTIONAL,
     required-properties               CHOICE {
        required-props                     [2] Font-Attribute-Set,
        rp-reference                       [3] External-Reference } OPTIONAL,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642846 687


```
      advisory-properties                CHOICE {
         advisory-props                      [4] Font-Attribute-Set,
         ap-reference                        [5] External-Reference } OPTIONAL,
      match-rules                            [6] Public-Object-ID OPTIONAL,
      satisfaction-criterion                 [7] Public-Object-ID OPTIONAL } }
--  Font Type 1 Font Specification
Glyph-Index-Map-ID              ::= CHOICE {
      glyph-index-map               [0] External-Resource-ID }
Fonttype1-Font-Spec             ::= [APPLICATION 25] IMPLICIT SEQUENCE {
      comment                           Comment OPTIONAL,
      font-spec-or-ref                  CHOICE {
         base-font-spec                     OCTET STRING,
         reference                          External-Reference } }
--  Remapped Font Specification
Remapped-Font-Spec              ::= [APPLICATION 26] IMPLICIT SEQUENCE {
      comment                           Comment OPTIONAL,
      font-id                       [0] External-Resource-ID,
      remap                             CHOICE {
         gim-id                         [1] External-Resource-ID,
         gitt-or-ref                    [2] CHOICE {
         gitt                                   Glyph-Index-Table
         reference                              External-Reference } } }
Glyph-Index-Table               ::= [APPLICATION 27] IMPLICIT SEQUENCE {
      comment                           Comment OPTIONAL,
      table-size                    [0]  IMPLICIT Cardinal,
      glyph-indices                 [1]  IMPLICIT SEQUENCE OF Cardinal }
--  Composite Font Specification
Composite-Font-Spec             ::= [APPLICATION 28] IMPLICIT SEQUENCE {
      comment                           Comment OPTIONAL,
      fmap-type                     [0]  IMPLICIT Cardinal,
      fmap-param                    [1]  FMap-Parameters,} OPTIONAL,
      font-index-map-or-ref         [2]  CHOICE {
         font-index-map                      Font-Index-Map,
         reference                           External-Reference },
      font-list                     [3]  IMPLICIT SEQUENCE OF Font-Spec }
FMap-Parameters                 ::= SET {
      comment                           Comment OPTIONAL,
      escchar                       [0]  IMPLICIT Cardinal OPTIONAL,
      shiftout                      [1]  IMPLICIT Cardinal OPTIONAL,
      shiftin                       [2]  IMPLICIT Cardinal OPTIONAL,
      subsvector                    [3]  IMPLICIT OCTETSTRING OPTIONAL }
Font-Index-Map                  ::= SEQUENCE {
      comment                           Comment OPTIONAL,
      map-size                      [0]  IMPLICIT Cardinal,
      index-list                    [1]  IMPLICIT SEQUENCE OF Cardinal }
--  User Font Specification
User-Font-Spec                  ::= [APPLICATION 29] IMPLICIT SEQUENCE {
      comment                           Comment OPTIONAL,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
      definition                        SEQUENCE OF CHOICE {
         tokensequence                       TokenSequence,
         reference                           External-Reference }
--  Glyph Index Map Specification
Glyph-Index-Map-Spec              ::= [APPLICATION 30] IMPLICIT SEQUENCE {
      comment                             Comment OPTIONAL,
      map-size                        [0] IMPLICIT Cardinal,
      glyph-id-list                   [1] IMPLICIT SEQUENCE OF Glyph-ID }
END
```

38.8 Document Production Instructions Module

```
ISO10180-SPDL-DPI  {iso(1) standard(0) 10180  modules(1) dpi(5)}
DEFINITIONS                       ::= BEGIN
EXPORTS                           Dimension, XYDimensions,
                                  Head-Locations, DPI-Declaration,
                                  Additional-DPI, Medium-Spec, Medium-Properties,
                                  Medium-Weight,  Medium-Pre-Punch,
                                  Medium-Multi-Part,  Medium-Finish-Edges,
                                  Medium-Labels, Additional-Property,
                                  Medium-Select-DPI,  Medium-Selection,
                                  Page-Identifier, Current-Medium-DPI,
                                  Copies-DPI, Page-Select-DPI,
                                  Page-Selection, Sup-Page-Select,  Sides-DPI,
                                  Plex-DPI, X-Image-Shift-DPI,
                                  Y-Image-Shift-DPI,  Current-Side-DPI,
                                  Finishing-DPI, Stitching-Spec, Binding-Spec,
                                  Trimming-Spec, Die-Cutting-Spec, Punching-Spec,
                                  Perforating-Spec, Slitting-Spec, Insert-Spec,
                                  Insert-Sheet,  Cover-Spec,
                                  Folding-Spec, Other-Finishing-Spec,
                                  Colorant-Set-DPI, Colorant-Set-Desc,
                                  Doc-Start-Msg-DPI,  Aux-Page-DPI,
                                  Doc-Comment-DPI, Timeout-DPI,
                                  Abort-Policy-DPI,  Stacking-DPI,
                                  Doc-End-Msg-DPI;
IMPORTS                           Cardinal, Number, Non-Negative-Number,
                                  Printable-String,  Environment-ID,
                                  Public-Object-ID
                                  FROM ISO10180-SPDL
                                  {iso(1) standard(0) 10180 modules(1) spdl(0)}
-- Useful data types
Dimension                        ::= SEQUENCE {
      comment                           Comment OPTIONAL,
      tolerance                     [0] Non-Negative-Number OPTIONAL,
      dimension-value                   CHOICE {
         named-dimension                [1] Environment-ID,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        numeric-dimension               [2] Non-Negative-Number } }
XYDimensions                    ::= SEQUENCE {
      comment                           Comment OPTIONAL,
      tolerance                     [0] Non-Negative-Number OPTIONAL,
      xydimensions-value                CHOICE {
         named-xydimensions             [1] Environment-ID,
         numeric-xydimensions           [2] IMPLICIT SEQUENCE    {
              x-dimension                     [0] Non-Negative-Number,
              y-dimension                     [1] Non-Negative-Number } } }
Head-Locations                  ::= SEQUENCE {
      comment                           Comment OPTIONAL,
      tolerance                     [0] Non-Negative-Number OPTIONAL,
      head-locations-value              CHOICE {
         named-head-locations           [1] Environment-ID,
         numeric-head-locations         [2] IMPLICIT SEQUENCE OF
                                              Non-Negative-Number } }
Edge                            ::= ENUMERATED {
                                      bottom-edge (0),
                                      right-edge (1),
                                      top-edge (2),
                                      left-edge (3) }
DPI-Declaration                 ::= [APPLICATION 31] IMPLICIT SET {
      comment                           Comment OPTIONAL,
      medium-dpi                    [0] IMPLICIT Medium-DPI OPTIONAL,
      medium-select-dpi             [1] IMPLICIT Medium-Select-DPI OPTIONAL
      current-medium-dpi            [2] IMPLICIT Current-Medium-DPI OPTIONAL,
      copies-dpi                    [3] IMPLICIT Copies-DPI OPTIONAL,
      page-select-dpi               [4] IMPLICIT Page-Select-DPI OPTIONAL,
      sup-page-select-dpi           [5] IMPLICIT Sup-Page-Select-DPI OPTIONAL,
      sides-dpi                     [6] IMPLICIT Sides-DPI OPTIONAL,
      plex-dpi                      [7] Plex-DPI OPTIONAL,
      x-image-shift-dpi             [8] X-Image-Shift-DPI OPTIONAL,
      y-image-shift-dpi             [9] Y-Image-Shift-DPI OPTIONAL,
      current-side-dpi              [10] IMPLICIT Current-Side-DPI OPTIONAL,
      finishing-dpi                 [11] IMPLICIT Finishing-DPI OPTIONAL,
      colorant-set-dpi              [12] IMPLICIT Colorant-Set-DPI OPTIONAL,
      doc-start-msg-dpi             [13] IMPLICIT Document-Start-Message-DPI OPTIONAL,
      aux-page-type-dpi             [14] IMPLICIT Aux-Page-Type-DPI OPTIONAL,
      doc-comment-dpi               [15] IMPLICIT Document-Comment-DPI OPTIONAL,
      timeout-dpi                   [16] IMPLICIT Timeout-DPI OPTIONAL,
      abort-policy-dpi              [17] Abort-Policy-DPI OPTIONAL,
      output-bin-dpi                [18] Output-Bin-DPI OPTIONAL,
      output-spec-dpi               [19] Output-Spec-DPI OPTIONAL,
      doc-end-msg-dpi               [20] IMPLICIT Document-End-Message-DPI OPTIONAL,
      additional-dpi                [21] IMPLICIT SET OF Additional-DPI OPTIONAL }
--  Medium DPI
Medium-DPI                      ::= SEQUENCE OF Medium-Declaration
Medium-Declaration              ::= SEQUENCE {
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642849 396


```
      comment                         Comment OPTIONAL,
      medium-id                  [0]  IMPLICIT Name,
      medium-spec                [1]  IMPLICIT Medium-Spec }
Medium-Spec                   ::= SET {
      comment                         Comment OPTIONAL,
      medium-name                [0]  Environment-ID OPTIONAL,
      medium-message             [1]  IMPLICIT Printable-String OPTIONAL,
      medium-props               [2]  IMPLICIT Medium-Properties OPTIONAL }
Medium-Properties             ::= SET {
      comment                         Comment OPTIONAL,
      medium-size                [0]  IMPLICIT XYDimensions OPTIONAL,
      medium-color               [1]  Environment-ID OPTIONAL,
      medium-weight              [2]  IMPLICIT Medium-Weight OPTIONAL,
      medium-type                [3]  Environment-ID OPTIONAL,
      medium-tooth               [4]  Environment-ID OPTIONAL,
      medium-grain               [5]  Environment-ID OPTIONAL,
      medium-pre-punch           [6]  IMPLICIT Medium-Pre-Punch OPTIONAL,
      medium-multi-part          [7]  IMPLICIT Medium-Multi-Part OPTIONAL,
      medium-finished-edges      [8]  IMPLICIT Medium-Finish-Edges OPTIONAL,
      medium-labels              [9]  IMPLICIT Medium-Labels OPTIONAL,
      additional-props           [10] IMPLICIT SET OF Additional-Property OPTIONAL }    ;
Medium-Weight                 ::= SET {
      comment                         Comment OPTIONAL,
      value                      [0]  IMPLICIT Cardinal,
      tolerance                  [1]  IMPLICIT Cardinal OPTIONAL }
Medium-Pre-Punch              ::= SET {
      comment                         Comment OPTIONAL,
      punch-count                [0]  IMPLICIT Cardinal,
      punch-edge                 [1]  IMPLICIT Edge OPTIONAL,
      punch-diameter             [2]  IMPLICIT Dimension OPTIONAL,
      punch-offset               [3]  IMPLICIT Dimension OPTIONAL,
      punch-locations            [4]  IMPLICIT Head-Locations OPTIONAL }
Medium-Multi-Part             ::= SET {
      comment                         Comment OPTIONAL,
      sequence-number            [0]  IMPLICIT Cardinal,
      sequence-length            [1]  IMPLICIT Cardinal }
Medium-Finish-Edges           ::= SET (SIZE (4)) OF SET {
      comment                         Comment OPTIONAL,
      finish-edge                [0]  IMPLICIT Edge,
      finish-type                [1]  Environment-ID OPTIONAL }
Medium-Labels                 ::= SET {
      comment                         Comment OPTIONAL,
      per-column                 [0]  IMPLICIT Cardinal,
      per-row                    [1]  IMPLICIT Cardinal }
Additional-Property           ::= SET {
      comment                         Comment OPTIONAL,
      property-name              [0]  Public-Object-ID,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642850 008


```
      property-value                [1]  ANY }  -- Defined by property-name --
--  Medium Select DPI
Medium-Select-DPI                ::= SEQUENCE OF Medium-Selection
Medium-Selection                 ::= SET {
      comment                           Comment OPTIONAL,
      start-page                  [0]   Page-Identifier,
      end-page                    [1]   Page-Identifier,
      medium-name                 [2]   Name }  -- Defined by property-name --
Page-Identifier                  ::= CHOICE { Cardinal }
Current-Medium-DPI               ::= Name
--  Presentation DPI
Copies-DPI                       ::= Cardinal
Page-Select-DPI                  ::= SEQUENCE OF Page-Selection
Sup-Page-Select-DPI              ::= SEQUENCE OF Page-Selection
Page-Selection                   ::= SEQUENCE {
      comment                           Comment OPTIONAL,
      start-page                  [0]   Page-Identifier,
      end-page                    [1]   Page-Identifier }
Sides-DPI                        ::= Sides
Plex-DPI                         ::= Public-Object-ID
X-Image-Shift-DPI                ::= Number
Y-Image-Shift-DPI                ::= Number
Current-Side-DP                  ::= Sides
Sides                            ::= Cardinal (1..2)
--  Finishing DPI
Finishing-DPI                    ::= SEQUENCE {
      comment                           Comment OPTIONAL,
      finishing-message           [0]   IMPLICIT Printable-String OPTIONAL,
      finishing-spec                    CHOICE {
         named-finishing          [1]   IMPLICIT SET {
              comment                         Comment OPTIONAL,
              finishing-id                    Environment-ID },
         finishing-spec-list      [2]   IMPLICIT SEQUENCE OF Finishing-Proc-Spec }
Finishing-Proc-Spec              ::= CHOICE {
      stitching-spec              [0]   Stitching-Spec,
      binding-spec                [1]   Binding-Spec,
      trimming-spec               [2]   Trimming-Spec,
      die-cutting-spec            [3]   Die-Cutting-Spec,
      punching-spec               [4]   Punching-Spec,
      perforating-spec            [5]   Perforating-Spec,
      slitting-spec               [6]   Slitting-Spec,
      insert-spec                 [7]   Insert-Spec,
      cover-spec                  [8]   Cover-Spec,
      folding-spec                [9]   Folding-Spec,
      other-finishing-spec        [10]  Other-Finishing-Spec }
Stitching-Spec                   ::= CHOICE {
      named-stitching             [0]   IMPLICIT SET {
         comment                              Comment OPTIONAL,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642851 T44


```
        stitching-id                    Environment-ID },
    parameters                  [1]  IMPLICIT SET {
        comment                          Comment OPTIONAL,
        reference-size              [0]  XYDimensions OPTIONAL,
        reference-edge              [1]  Edge OPTIONAL,
        jog-edge                    [2]  Edge OPTIONAL,
        process-offset              [3]  Dimension OPTIONAL,
        head-locations              [4]  Head-Locations OPTIONAL,
        stitch-type                 [5]  Environment-ID OPTIONAL } }
Binding-Spec                ::= CHOICE {
    named-binding               [0] IMPLICIT SET {
        comment                          Comment OPTIONAL,
        binding-id                       Environment-ID },
    parameters                  [1] IMPLICIT SET {
        comment                          Comment OPTIONAL,
        reference-size              [0]  XYDimensions OPTIONAL,
        reference-edge              [1]  Edge OPTIONAL,
        jog-edge                    [2]  Edge OPTIONAL,
        binding-type                [3]  Environment-ID OPTIONAL,
        binding-color               [4]  Environment-ID OPTIONAL } }
Trimming-Spec               ::= CHOICE {
    named-trimming              [0] IMPLICIT SET {
        comment                          Comment OPTIONAL,
        trimming-id                      Environment-ID },
    parameters                  [1] IMPLICIT SET {
        comment                          Comment OPTIONAL,
        reference-size              [0]  XYDimensions OPTIONAL,
        reference-edge              [1]  Edge OPTIONAL,
        jog-edge                    [2]  Edge OPTIONAL,
        trim-offset                 [3]  Dimension,
        trim-dimensions             [4]  XYDimensions } }
Die-Cutting-Spec            ::= CHOICE {
    named-die-cutting           [0] IMPLICIT SET {
        comment                          Comment OPTIONAL,
        die-cutting-id                   Environment-ID },
    parameters                  [1] IMPLICIT SET {
        comment                          Comment OPTIONAL,
    reference-size                  [0]  XYDimensions OPTIONAL,
        reference-edge              [1]  Edge OPTIONAL,
        jog-edge                    [2]  Edge OPTIONAL,
        die-cut-position            [5]  XYDimensions,
        die-cut-name                [6]  Environment-ID } }
Punching-Spec               ::= CHOICE {
    named-punching              [0] IMPLICIT SET {
        comment                          Comment OPTIONAL,
        punching-id                      Environment-ID },
    parameters                  [1] IMPLICIT SET {
        comment                          Comment OPTIONAL,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        reference-size              [0]  XYDimensions OPTIONAL,
        reference-edge              [1]  Edge OPTIONAL,
        jog-edge                    [2]  Edge OPTIONAL,
        process-offset              [3]  Dimension OPTIONAL,
        head-locations              [4]  Head-Locations OPTIONAL,
        punch-diameter              [5]  Dimension OPTIONAL } }
Perforating-Spec            ::= CHOICE {
    named-perforating           [0] IMPLICIT SET {
        comment                         Comment OPTIONAL,
        perforating-id                  Environment-ID },
    parameters                  [1] IMPLICIT SET {
        comment                         Comment OPTIONAL,
        reference-size              [0]  XYDimensions OPTIONAL,
        reference-edge              [1]  Edge OPTIONAL,
        jog-edge                    [2]  Edge OPTIONAL,
        process-offset              [3]  Dimension,
        head-locations              [4]  Head-Locations,
        perforation-type            [5]  Environment-ID } }
Slitting-Spec               ::= CHOICE {
    named-slitting              [0] IMPLICIT SET {
        comment                         Comment OPTIONAL,
        slitting-id                     Environment-ID },
    parameters                  [1] IMPLICIT SET {
        comment                         Comment OPTIONAL,
        reference-size              [0]  XYDimensions OPTIONAL,
        reference-edge              [1]  Edge OPTIONAL,
        jog-edge                    [2]  Edge OPTIONAL,
        head-locations              [4]  Head-Locations} }
Insert-Spec                 ::= CHOICE {
    named-inserting             [0] IMPLICIT SET {
        comment                         Comment OPTIONAL,
        inserting-id                    Environment-ID },
    parameters                  [1] IMPLICIT SET {
        comment                         Comment OPTIONAL,
        reference-edge              [1]  Edge OPTIONAL,
        jog-edge                    [2]  Edge OPTIONAL,
        insert-sheet-list           [5]  IMPLICIT SEQUENCE OF Insert-Sheet } }
Insert-Sheet                ::= SET {
    comment                         Comment OPTIONAL,
    insert-id                       CHOICE {
        insert-name                 [0] Environment-ID,
        insert-bin                  [1] Cardinal },
    insert-top-surface          [2] ENUMERATED {
                                        top (0),
                                        bottom (1) } OPTIONAL,
    insert-after                [3] Cardinal,
    insert-message              [4] Printable-String OPT    IONAL }
Cover-Spec                  ::= CHOICE {
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

4851903 0642853 817


```
      named-covers                  [0] Environment-ID,
      parameters                    [1] IMPLICIT SET {
         comment                            Comment OPTIONAL,
         reference-size                 [0]  XYDimensions OPTIONAL,
         reference-edge                 [1]  Edge OPTIONAL,
         jog-edge                       [2]  Edge OPTIONAL,
         front-cover                    [5]  CHOICE {
              cover-name                     [0] Environment-ID,
              cover-spec                     [1] Medium-Description } OPTIONAL,
         back-cover                     [6]  CHOICE {
              cover-name                     [0] Environment-ID,
              cover-spec                     [1] Medium-Description } OPTIONAL }
Folding-Spec                    ::= CHOICE {
      named-folding                 [0] IMPLICIT SET {
         comment                            Comment OPTIONAL,
         folding-id                         Environment-ID },
      parameters                    [1] IMPLICIT SET {
         comment                            Comment OPTIONAL,
         reference-size                 [0]  XYDimensions OPTIONAL
         reference-edge                 [1]  Edge OPTIONAL,
         jog-edge                       [2]  Edge OPTIONAL,
         head-locations                 [4]  Locations} }
Other-Finishing-Spec            ::= SET {
      finishing-op-name             [0] Public-Object-ID,
      finishing-op-parameters               CHOICE {
         named-op                           [1] IMPLICIT SET {
              comment                            Comment OPTIONAL,
              other-finishing-id                 Environment-ID },
         parameters                         [2] IMPLICIT SET {
              comment                                Comment OPTIONAL,
              reference-size                     [0]  XYDimensions OPTIONAL,
              reference-edge                     [1]  Edge OPTIONAL,
              jog-edge                           [2]  Edge OPTIONAL,
              process-offset                     [3]  Dimension OPTIONAL,
              head-locations                     [4]  Locations OPTIONAL,
              other-parameters                   [5]  ANY } } }
--  Colorant Set DPI
Colorant-Set-DPI                ::= SEQUENCE {
      comment                           Comment OPTIONAL,
      message                       [0]  Printable-String OPTIONAL,
      colorant-set-spec                 CHOICE {
         colorant-set-id                [1] Environment-ID,
         colorant-set-desc              [2] IMPLICIT Colorant-Set-Desc } }
Colorant-Set-Desc               ::= SEQUENCE {
      comment                           Comment OPTIONAL,
      colorant-set-class            [0]  Public-Object-ID,
      colorant-spec                 [1]  IMPLICIT SEQUENCE OF Colorant-ID }
Colorant-ID                     ::= Environment-ID
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642854 753


```
Doc-Start-Msg-DPI              ::= Printable-String
Aux-Page-Type-DPI              ::= SET {
     comment                          Comment OPTIONAL,
     start-page                  [0]  Environment-ID OPTIONAL,
     separator-page              [1]  Environment-ID OPTIONAL,
     end-page                    [2]  Environment-ID OPTIONAL }
Doc-Comment-DPI                ::= Printable-String
Timeout-DPI                    ::= Cardinal
Abort-Policy-DPI               ::= Environment-ID
Output-Bin-DPI                 ::= SET {
     comment                          Comment OPTIONAL,
     bin-identifier                   CHOICE {
        bin-name                      [0] Environment-ID,
        bin-number                    [1] IMPLICIT Cardinal } OPTIONAL,
     output-positions            [2] IMPLICIT Cardinal OPTIONAL }
Output-Spec-DPI                ::= SET {
     comment                          Comment OPTIONAL,
     collated                    [0] IMPLICIT BOOLEAN OPTIONAL,
     offset                      [1] IMPLICIT BOOLEAN OPTIONAL,
     burst                       [2] IMPLICIT BOOLEAN OPTIONAL,
     additional                  [3] IMPLICIT SET OF SET {
          output-spec-name            [0] Public-Object-ID,
          output-spec-value           [1] ANY }
                                           -- Defined by output-spec-name --
          output-spec-name            OPTIONAL }
Doc-End-Msg-DPI                ::= Printable-String
--  Additional DPI
Additional-DPI                 ::= [APPLICATION 32] IMPLICIT SEQUENCE{
     comment                          Comment OPTIONAL,
     dpi-name                    [0] Public-Object-ID,
     dpi-value                   [1] ANY }  -- Defined by dpi-name --
END
```

38.9 Binary Content Representation and Interchange Format

The SPDL content consists of sequences of *tokens*. Each token represents a unit of information which is processed by the content processor. This clause specifies the binary encoding of the tokens in SPDL content.

38.9.1 Token structure and notation

Each token is encoded by a sequence of one or more octets. Each octet consists of eight binary bits which may be specified in any of three ways:

— by specification of the individual bit values, with the high-order bit referred to as bit 8 and the low order bit as bit 1;

— as a sequence of bit values in which the first or leftmost bit value is the value is the highest order or most significant bit in the octet (bit 8);

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— by the numerical value (between 0 and 255) obtained by considering the above sequence of bit values as binary digits.

The numerical value is referred to as the *binary value* of the octet. See figure 14.

38.9.1.1 Fields

Each token consists of one or more sequences of bits which specifies the token type, length, or value. Each such sequence of bits is called a *field*. A field may be contained in a single octet or may extend over several octets. The fields which specify the token type, length, and value are called the *type field*, the *length field* and the *value field* respectively.

38.9.1.2 Binary Value of a Field

The *binary value of a field* is the numerical value represented by the bits in that field considered as binary digits. If the field consists of the bits of more than one octet, the bits of the first octet of the field shall be the high order bits in the field. Thus if the field consists of n bits b_n . . b_1, the binary value of the field shall be

$$b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \ldots b_1 \times 2^0$$

38.9.1.3 Signed Integer value field

A *signed integer value of a field* depends on the value of the high order bit of the first octet of the field

— If the high order bit of the field is zero, the signed integer value of the field shall be the binary value of the field.

— If the high order bit of the first octet is one and the length of the field is n bits, the signed integer value of the field shall be the binary value of the field minus 2^n.

NOTE 232 This is the two's complement representation of the signed integer value.

38.9.2 Token types

The binary content encoding uses four basic token types:

— Short Opcode

— Type/Value

— Type/Length/Value

— Short Integer

The formats of each of basic token types, together with the bit values which distinguish them, are shown in figure 15

The Type/Value and Type/Length/Value types are further subdivided into 32 subtypes each. For these tokens the type field consists of the basic type specification and the subtype specification.

The first octet of each token specifies the token type and is called the type octet. The basic token type is determined by the value of the type octet as shown in table 39.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

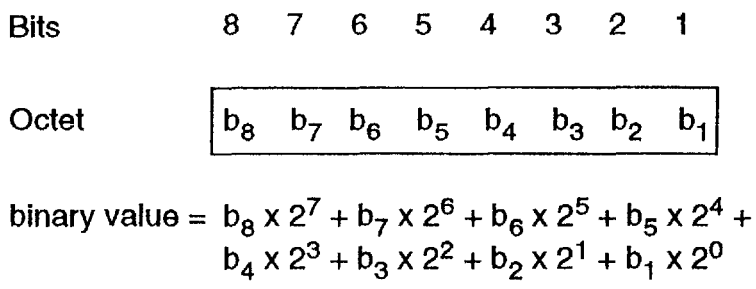

Figure 14 — Bit Numbering and Binary Value of an Octet

Table 39

Type Octet	Short Opcode token
64-95	Type/Value token
96-127	Type/Length/Value token
128-255	Short Integer token

The type octet also contains additional information which depends on the basic token type.

- For Short Opcode tokens the type octet contains the value field as well as the type field.
- For Type/Value and Type/Length/Value tokens the subtype specification is part of the type field and is contained in the type octet
- For Short Integer tokens the type octet contains the high order 7 bits of the value field as well as the type field.

The length of Short Opcode, Type/Value and Short Integer tokens is implied by the token type. The length of the type field and the length field of a Type/Length/Value token are implied by the token type, while the number of octets which comprise the value field of is contained in the length field

The binary encoding provides alternate methods for several varieties of tokens. Any Octet String or Data Block Type/Length/Value token is equivalent to a Long Octet String or Long Data Block Type/Length/Value token with the same value; any Short Integer token is equivalent to a Type/Value Integer token with the same value; and any Short Opcode token is equivalent to a Type/Length/Value executable name token whose value is the Name assigned to the value of the Short Opcode value

38.9.3 Representation of values

Binary values are used to distinguish token types and subtypes and to specify the length of the value fields of Type/Length/Value tokens. The *binary value* represented by a field shall be the value of the binary number whose digits are the sequence of bits in the field which are contained in the first octet, followed by the sequence of bits in the field which are contained in the second and any following octets.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

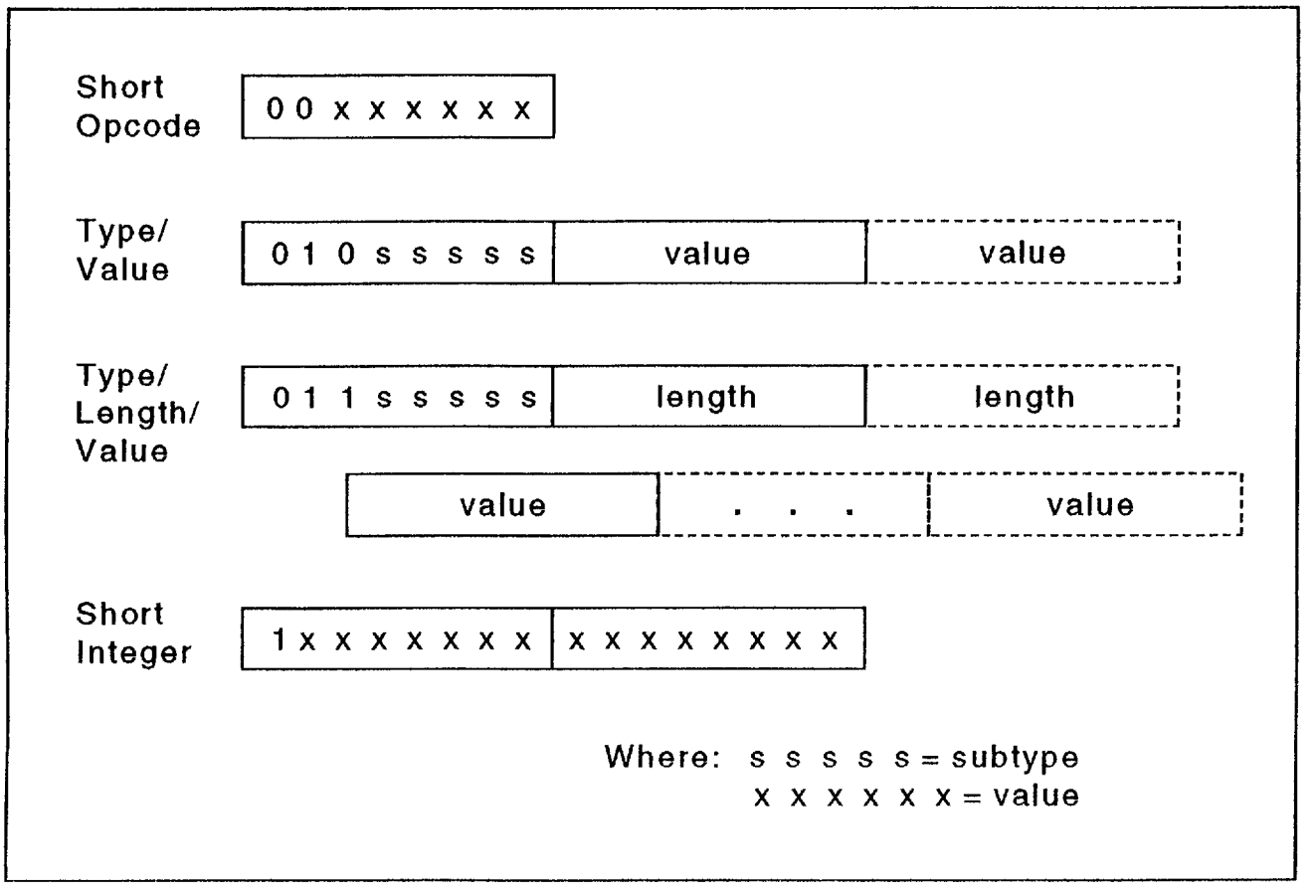

Figure 15 — Basic Token Types

38.9.3.1 Unsigned Integer value field

An *Unsigned Integer value field* shall consist of either two or four octets. The value of the integer represented shall be the binary value of the Unsigned Integer value field.

38.9.3.2 Signed Integer value field

A *Signed Integer value field* shall consist of either two or four octets. The value of the integer represented shall be the signed integer value of the Signed Integer value field.

38.9.3.3 Floating Point Real Number value field

A *Floating Point Real Number value field* shall consist of four octets. The value of the real number shall be represented as a basic single width floating point binary number as specified by IEEE 754.

38.9.3.4 Fixed Point Real Number value field

A *Fixed Point Real Number value field* shall consist of either three or five octets. If the binary value of the first octet in the value field is r and the signed integer value of of the field consisting of the remaining two or four octets in the value field is n, then the value of the real number represented shall be $n \div 2^r$.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

38.9.3.5 Octet String value field

An *Octet String value field* shall consist of a variable number of octets. The value represented shall be the sequence of octets which comprise the value field.

38.9.3.6 Data Block value field

A *Data Block value field* shall consist of a variable number of octets. The value represented shall be the sequence of octets which comprise the value field.

38.9.3.7 Procedure value field

A *Procedure value field* shall consist of a variable number of octets. The value represented shall be the sequence of tokens (in the binary encoding) represented by the sequence of octets which comprises the the value field.

38.9.3.8 Vector of homogeneous numbers value field

A *Vector of homogeneous numbers value field* shall consist of a variable number of octets.

— The binary value of the first octet of the Vector of homogeneous numbers value field shall be the number of octets used to represent each element of the Vector.

— The value of the second octet of the Vector of homogeneous numbers value field shall specify the form in which the numbers are represented.

— The remainder of the value field shall consist of an integral number of sub-fields each of which represents a number

The Vector represented by the Vector of homogeneous numbers value field shall be the Vector formed by these numbers.

The representation of the elements of the Vector shall depend on the value of the 2nd octet as shown in table 40

Table 40

2nd Octet	Number representation
0	Unsigned integer as represented by an Unsigned Integer value field
1	Signed integer as represented by a Signed Integer value field
2	Floating Point Real Number as represented by a Floating Point Real Number value field
>8	Fixed point Real Number as represented by the second and subsequent octets of a Fixed Point Real Number value field whose initial octet is the value of the 2nd octet minus 8

38.9.4 Short Opcode tokens

A Short Opcode token represents one of 64 executable Names in a single octet. A Short Opcode token is constructed as follows:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— bit 8 and bit 7 of the type octet are zero;

— bit 6 through bit 1 of the type octet constitute the value field.

The executable Name represented by the Short Opcode token is the Name which corresponds to the binary value of the value field, as specified in annex A.

38.9.5 Type/Value tokens

There are 32 subtypes of the Type/Value token type. The Type/Value type is identified by the high order 3 bits of the type octet, while the subtype is specified by the low order 5 bits of the type octet.

The length of the token and the type of the value represented are implicit in the token type and subtype. The type octet is followed by one or more octets which constitute the value field. The number of octets in the value field and the value specified by the value field is shown in table 41.

Table 41

Type Octet	Value Length	Value Specified by the Value Field
64	1	The executable Name corresponding to the binary value of the value field, as specified in annex a
65	1	The executable Name corresponding to 256 plus the binary value of the value field, as specified in annex a
66-67	-	Reserved
68	2	The Signed Integer represented by the Signed Integer value field consisting of the two octets which follow the type octet
69	4	The Signed Integer represented by the Signed Integer value field consisting of the four octets which follow the type octet
70	4	The Real Number represented by the Floating Point Real Number value field consisting of the four octets following the type octet
71	3	The Real Number represented by the Fixed Point Real Number value field consisting of the three octets following the type octet
72	5	The Real Number represented by the Fixed Point Real Number value field consisting of the five octets following the type octet
73-95	-	Not assigned

NOTES

233 The value represented by the value field of tokens whose type octet value is 64 or 65 is the binary value of the field consisting of the low order 13 bits of the token.

234 Subtypes corresponding to type octet values of 66 and 67 are reserved for additional executable Name values.

38.9.6 Type/Length/Value tokens

Type/Length/Value tokens are variable length tokens. The type octet shall be followed by one, two, or four octets which specify the number of octets in the value field. Together these octets are called the *length field*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642860 T57

The type octet values, the length of the corresponding length field, and the type and encoding of the value are shown in table 42.

Table 42

Type Octet Value	Length Field (octets)	Value Specified by the Value Field
96	1	an executable Name as represented in the clear text encoding
97	1	a literal Name as represented in the clear text encoding
98	1	an Octet String
99	2	an Octet String
100	2	a Data Block
101	4	a Data Block
102	2	an incomplete Data Block. The following token must be either a Data Block token (type octet = 100 or 101) or another incomplete Data Block token (type octet = 102)
103	2	a Procedure
104	2	a Vector of homogeneous numbers
105-126		unassigned
127	2	a sequence of octets representing a sequence of encrypted tokens. The first two octets of the value field contain an encryption identifier. The remaining octets contain an encrypted representation of a sequence of tokens

38.9.7 Short Integer tokens

A short integer token represents an Integer in the range of -4096 . . . 28 671. The value of the integer shall be the binary value of the type octet minus 36 864.

38.9.8 Encoding values for operators

The operators which are defined by the SPDL syntax can be represented by Type/Length/Value tokens whose value is the Name of the operator or by tokens which identify the operator by means of an encoding value or *opcode*. The opcodes which represent the SPDL operators are specified in annex A

39 Conformance

39.1 SPDL Instances

A conforming SPDL Instance is one whose syntax conforms to clause 5 through clause 35, and whose representation conforms to clause 37 or to clause 38.

Every conforming DOCUMENT shall have at least one PAGESET or PAGE structure element.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642861 993

39.2 SPDL Presentation Processes

This International Standard specifies the effects of processing an SPDL Instance It does not constrain the implementation of an SPDL Presentation Process beyond the requirement that the specified effects be produced.

Every SPDL Presentation Process will have limitations (e.g. on storage capacity) which restrict the complexity and size of SPDL Instances which can be processed. Every conforming SPDL Presentation Process shall be able to process any conforming SPDL Instance within these limitations. An attempt to process an SPDL Instance which results in the exceeding of one of these limitations may result in a structure error and incomplete or aborted processing.

Notwithstanding these limitations, every conforming SPDL Presentation Process shall be able to:

— define resources and add them to the environment of the Presentation Process as specified in 39.5.1;

— construct and save objects in content processing as specified in 39.8 1;

— represent values in content processing as specified in 39.8.2.

39.3 Representation and Interchange Format

Every conforming SPDL Presentation Process shall either:

— be able to process all conforming SPDL Instances in the Clear Text Representation and Interchange Format (as specified in 37.2);

— be able to process all conforming SPDL Instances in the Binary Representation and Interchange Format (as specified by 38.9);

— be able to process all conforming SPDL Instances in either the Clear Text Representation and Interchange Format or the Binary Representation and Interchange Format.

The statements in 39.4 through 39.9 apply only to SPDL Instances in a Representation and Interchange Format which can be processed by the particular SPDL Presentation Process.

39.4 Structure Syntax

Every conforming SPDL Presentation Process shall be able to process the structure of any conforming SPDL Instance as specified by clause 5 through clause 16.

39.5 Resources

39.5.1 Resource Definitions which are Subordinate to an ENVIRONMENT RESOURCE

A conforming Presentation Process need not add to the environment of the Presentation Process a resource which is defined by a RESOURCE DEFINITION which is subordinate to an ENVIRONMENT RESOURCE, or to remove from the environment of the Presentation Process a resource which is undefined by a subordinate RESOURCE UNDEFINITION structure element. However, even if no resource is added to or removed from the environment of the Presentation Process, ENVIRONMENT RESOURCE processing shall conform to 9.2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642862 82T

39.5.2 Resource Definitions which are Subordinate to a DOCUMENT

Every conforming Presentation Process shall be able to process RESOURCE DEFINITION structure elements which are subordinate to a DOCUMENT as specified in clause 9 and clause 10, and to make the resources defined by them available to content processing, within the limitations specified in 39.2 .

Every conforming SPDL Presentation Process shall have the ability to create each of the resources specified below and add it to the environment of the Presentation Process for the scope of the defining RESOURCE DEFINITION:

- a Dictionary resource defined by a DICTIONARY SPECIFICATION for which the content element which is obtained by concatenating the values of the subordinate TOKENSEQUENCE structure elements contains no more than than 65 535 octets;

- a Color Space resource defined by a COLOR SPACE SPECIFICATION for which the content element which is obtained by concatenating the values of the subordinate TOKENSEQUENCE structure elements contains no more than than 65 535 octets;

- a Data Stream resource defined by a DATA SOURCE SPECIFICATION for which the value of the subordinate DATA BLOCK consists of an Octet String of no more than than 65 535 octets;

- a Pattern resource defined by a PATTERN SPECIFICATION for which the content element which is obtained by concatenating the values of the subordinate TOKENSEQUENCE structure elements contains no more than than 65 535 octets;

- a Form resource defined by a FORM SPECIFICATION for which the content element which is obtained by concatenating the values of the subordinate TOKENSEQUENCE structure elements contains no more than than 65 535 octets;

- an Indexed Font resource defined by a FONTTYPE1 FONT SPECIFICATION for which the value contains no more than than 65 535 octets;

- an Indexed Font resource defined by a USER FONT SPECIFICATION for which the content element which is obtained by concatenating the values of the subordinate TOKENSEQUENCE structure elements contains no more than than 65 535 octets.

These requirements are separate: a conforming SPDL Presentation Process need not be able to include more than one such resource in the environment of the Presentation Process at any one time.

39.5.3 Resources in the Environment of the Presentation Process

Every conforming SPDL Presentation Process shall have in its environment a resource of the identified resource class which is identified by each of the Public Object Identifiers listed below and which conforms to the applicable specification in this International Standard. The resource classes and the Object Names corresponding to these Public Object Identifier values are listed in 39 5 3.1 through 39.5 3.5.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642863 766

39.5.3.1 Color Spaces

Table 43

Resource Class Identifier	Object Name
ColorSpace	*ColorSpace::DeviceRGB*
ColorSpace	*ColorSpace::DeviceCMYK*
ColorSpace	*ColorSpace::DeviceKX*
ColorSpace	*ColorSpace::DeviceGray*

39.5.3.2 Data Sources

Table 44

Resource Class Identifier	Object Name
DataSource	*DataSource::Document*

39.5.3.3 Filters

Table 45

Resource Class Identifier	Object Name
Filter	*Filter::ASCIIHexDecode*
Filter	*Filter::ASCII85Decode*
Filter	*Filter::RunLengthDecode*
Filter	*Filter::CCITTFaxDecode*
Filter	*Filter::NullDecode*

39.5.3.4 Indexed Font Resources

Table 46

Resource Class Identifier	Object Name
IndexedFont	*Fonts::ISO-Serif::Regular*
IndexedFont	*Fonts::ISO-Serif::Bold*
IndexedFont	*Fonts.:ISO-Serif::Italic*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642864 6T2

Table 46 (concluded)

Resource Class Identifier	Object Name
IndexedFont	*Fonts::ISO-Serif::BoldItalic*
IndexedFont	*Fonts::ISO-SanSerif::Regular*
IndexedFont	*Fonts::ISO-SanSerif::Bold*
IndexedFont	*Fonts::ISO-SanSerif::Italic*
IndexedFont	*Fonts::ISO-SanSerif::BoldItalic*
IndexedFont	*Fonts::ISO-Monospace::Regular*
IndexedFont	*Fonts::ISO-Monospace::Bold*
IndexedFont	*Fonts::ISO-Monospace::Italic*
IndexedFont	*Fonts::ISO-Monospace::BoldItalic*

39.5.3.5 Glyph Index Maps

Table 47

Resource Class Identifier	Object Name
GlyphIndexMap	*GlyphIndexMap::Latin 1 Publishing*
GlyphIndexMap	*GlyphIndexMap::IR::nnnn*

Where "*nnnn*" is 0 or any sequence of decimal digits beginning with a non-zero digit.

39.5.4 Color Space Families

Every conforming SPDL Presentation Process shall be capable of supporting the following Color Space Families

Table 48

Color Space Family	Public Identifier (canonical character string form)
CIELAB	*ISO/IEC 10180//ColorSpaceFamily. CIELAB*
CIELUV	*ISO/IEC 10180//ColorSpaceFamily::CIELUV*
CIEBasedABC	*ISO/IEC 10180//ColorSpaceFamily: CIEBasedABC*
CIEBasedA	*ISO/IEC 10180//ColorSpaceFamily::CIEBasedA*
DeviceRGB	*ISO/IEC 10180//ColorSpaceFamily::DeviceRGB*
DeviceCMYK	*ISO/IEC 10180//ColorSpaceFamily::DeviceCMYK*
DeviceGray	*ISO/IEC 10180//ColorSpaceFamily::DeviceGray*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642865 539

Table 48 (concluded)

Color Space Family	Public Identifier (canonical character string form)
Indexed	*ISO/IEC 10180//ColorSpaceFamily::Indexed*
NamedColor	*ISO/IEC 10180//ColorSpaceFamily::NamedColor*

Every conforming SPDL Presentation Process shall have a Color Space Family Identifier which which identifies each of the above Color Space Families. The content of the OctetString referenced by each of the Color Space Family Identifiers shall be as specified in 9.6.1.

Every conforming SPDL Presentation Process shall be able to process COLOR SPACE SPECIFICATION structure elements which specify Color Space Objects for each of the above Color Space Families, and shall provide a default mapping from each such Color Space to the gamut of the imaging device.

39.6 External Structure Elements

Every conforming SPDL Presentation Process shall have associated with it a domain, possibly null, from which external structure elements may be obtained. There need not be any external structure elements in this domain.

39.7 Document Production Instructions

Every conforming SPDL Presentation Process shall support all of the Document Production Instructions defined in clause 11 through clause 15, either by supporting the defined semantics or by the specified fallback behavior. For those Document Production Instructions for which no fallback is specified, conforming Presentation Processes shall implement the specified semantics.

39.8 Content Processing

Every Presentation Process will have limits which restrict the complexity and size of documents which can be processed. Within these limitations, a conforming Presentation Process must be able to process all content elements of any conforming SPDL instance.

Every conforming SPDL Presentation Process shall have the following minimum capabilities for content processing.

39.8.1 Ability to construct and save objects

Every conforming SPDL Presentation Process shall have the ability to:

— store a sequence of 511 Integers on the operand stack;

— construct a Vector of 8192 Integers;

— store a String representing 255 Glyphs;

— construct a Dictionary containing 1023 name/value pairs for which the value is an Integer;

— construct a Path containing 1500 path elements created by a sequence of tokens of the form {*BeginPathSegment LineTo LineTo . . . LineTo*};

— store 20 *DictionaryReferences* on the Context Stack;

— store 15 State Save levels;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642866 475

-- store 30 Graphics State Save Levels (including those saved as a result of global **SaveState** operations;

These requirements are separate: a conforming SPDL Presentation Process need not be able to store more than one of the above data objects concurrently.

39.8.2 Ability to represent values

Every conforming SPDL Presentation Process shall have the capability of representing at least the following values for content processing.

-- Integer values as specified by 6 3 for a value of *MaxInteger* $v2^{31}$ -1 and a value of *MinIntegerσ* - $2^{31}+1$;

— Real Number values which can be represented as basic single width floating point binary numbers as specified by IEEE 754:1987;

— Octet Strings of minimum length 65 535 octets.

39.9 Exception Handling

Every conforming SPDL Presentation Process shall provide exception handling as specified in clause 36.

39.10 Imaging

The effects of imaging operations are defined in terms or ideal results. Every SPDL Presentation Process will have limits on the ability to achieve the specified results. This International Standard does not specify requirements for the ability to achieve the specified imaging results

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642867 301

© ISO/IEC

Annex A
(normative)

Operator Encodings

All SPDL Operators are associated with Names in *SystemDict*. In the cleartext encoding of the Content Interchange Format, there is no explicit encoding for objects of type Operator; the Names of the Operators in *SystemDict* are instead encoded and may then be used to execute the Operators to which they are bound, either implicitly or explicitly, in content.

In the binary encoding of the Interchange Format, the Names of the Operators in *SystemDict* may be represented either by *Type/Length/Value* tokens whose value is the Name, or by means of an encoding value or opcode.

The following table lists the SPDL Operators, their Names in *SystemDict* for use in the cleartext or binary Content Interchange Format, and their opcodes in the binary Content Interchange Format.

NOTE 235 The additional Names associated with the Operator **Mark** are convenience notations for use in the cleartext Content Interchange Format with the **MakeandStoreVector** and **MakeandStoreDictionary** operators, respectively. See 37 2

Table A.1

Operator	Name	Opcode
AbsoluteValue	abs	60
Add	add	49
AnchorSearch	anchorsearch	61
And	and	62
AppendPath	uappend	63
ArcTangent	atan	64
ArcToClockwise	arcn	32
ArcToCounterClockwise	arc	31
BeginPathSegment	moveto	18
BeginPathSegmentRelative	rmoveto	19
Capacity	capacity	65
Ceiling	top	66
CheckIfExecutable	echeck	67
CheckIfReadable	rcheck	68
CheckIfWriteable	wcheck	69

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642868 248

Operator	Name	Opcode
ClearStack	clear	70
ClearToMark	cleartomark	71
ClipPath	clip	27
ClipPathEvenOdd	eoclip	28
ClipPathSegment	closepath	35
Concat	concat	72
ConcatT	concatmatrices	13
ContextStack	dictstack	73
ConvertToExecutable	cve	74
ConvertToIdentifier	cvid	75
ConvertToInteger	cvi	76
ConvertToReal	cvr	77
ConvertToString	cvs	78
Copy	copy	78
Cosine	cos	80
Count	count	81
CountToMark	counttomark	82
CurveTo	curveto	33
CurveToRelative	rcurveto	34
Define	def	7
DefineFont	deffontobj	83
Divide	div	84
Dup	dup	2
EntriesUsed	entriesused	85
Equal	eq	46
Exchange	exch	86
Execute	exec	87
ExecuteForm	execform	88
ExecuteTrapped	stopped	89
Exit	exit	53

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642869 184

Operator	Name	Opcode
Exponentiate	exp	90
False	false	45
FillPath	fill	42
FillPathEvenOdd	eofill	43
Filter	filter	91
FindFont	findfontobj	92
FindResource	findres	93
Floor	bottom	94
For	for	54
ForAll	forall	55
Get	get	3
GetBlackGeneration	currentblackgeneration	95
GetColor	currentcolor	15
GetColorRendering	currentcolorrendering	96
GetColorSpace	getcolorspace	36
GetCurrentDictionary	currentdict	97
GetDashPattern	currentdash	98
GetDeviceDescription	getdevinfo	99
GetHalftone	currenthalftone	100
GetInterval	getinterval	101
GetMiterLimit	currentmiterlimit	102
GetOverPrint	currentoverprint	103
GetPath	getpath	26
GetPosition	currentpoint	20
GetRootFont	currentfont	17
GetSelectedFont	rootfont	104
GetStrokeAdjust	currentstrokeadjust	105
GetStrokeEnd	curentlinecap	106
GetStrokeJoin	currentlinejoin	107
GetStrokeWidth	currentlinewidth	108

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642870 9T6

Operator	Name	Opcode
GetTest	known	4
GetTrans	gettrans	109
GetUnderColorRemoval	currentundercolorremoval	111
GetValue	load	5
GetValueTest	where	6
GlyphToPath	charpath	112
GreaterOrEqual	GE	113
GreaterThan	GT	114
If	if	47
IfElse	ifelse	48
ImageRasterElement	image	115
Index	index	116
IntegerDivide	idiv	117
LessOrEqual	LE	118
LessThan	LT	119
LineTo	lineto	29
LineToRelative	rlineto	30
Logarithm	log	120
LogicalShift	bitshift	121
Loop	loop	56
MakeDictionary	dict	122
MakeExecuteOnly	executeonly	123
MakePattern	makepattern	124
MakeReadOnly	readonly	125
MakeString	string	126
MakeVector	array	127
MakeandStoreDictionary	>>	128
MakeandStoreVector	]	129
Mark	mark	8
Mark	[	8

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642871 832

Operator	Name	Opcode
Mark	<<	8
MaskBitMap	imagemask	130
Multiply	mul	131
NaturalLogarithm	ln	132
Negate	neg	133
NewPath	newpath	24
Noop	noop	1
Not	not	134
NotEqual	ne	51
Null	null	52
OpenFont	openfont	135
Or	or	136
OutlineStroke	strokepath	
Pop	pop	137
PopContextStack	end	138
Print	print	139
PushContextStack	begin	140
Put	put	9
PutInterval	putinterval	142
PutValue	store	10
PutWMode	putWMode	143
QueryResource	queryresource	193
RaiseError	raiseerror	144
RaiseException	stop	145
RaiseWarning	raisewarning	146
Rand	rand	147
RandSetState	srand	148
RectangleClip	rectclip	189
RectangleFill	rectfill	190
RectangleStroke	rectstroke	191

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642872 779

Operator	Name	Opcode
Remainder	mod	149
Repeat	repeat	57
RestoreGraphicState	grestore	59
RestoreGraphicStateXCP	grestoreXCP	150
RestoreSavedGraphicState	grestoreall	151
RestoreState	restore	152
Roll	roll	153
Rotate	rotate	141
RotateT	rotateT	154
Round	rnd	155
SaveGraphicsState	gsave	58
SaveState	save	156
Scale	scale	11
ScaleT	scaleT	187
ScaleFont	scalefont	157
Search	search	158
SetBlackGeneration	setblackgeneration	159
SetColor	setsolidcolor	14
SetColorRendering	setcolorrendering	160
SetColorSpace	setcolorspace	163
SetDashPattern	setdash	40
SetFont	setfont	16
SetHalftone	sethalftone	166
SetMiterLimit	setmiterlimit	167
SetOverPrint	setoverprint	168
SetPath	setpath	25
SetPatternColor	setpatterncolor	169
SetPosition	moveto	18
SetPositionRelative	rmoveto	19
SetStrokeAdjust	setstrokeadjust	173

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642873 605

Operator	Name	Opcode
SetStrokeEnd	setlinecap	39
SetStrokeJoin	setlinejoin	38
SetStrokeWidth	setlinewidth	37
SetTrans	settrans	174
SetUnderColorRemoval	setundercolorremoval	176
ShowGlyph	showglyph	21
ShowString	show	22
ShowStringEscapedX	xshow	23
ShowStringEscapedXY	xyshow	177
ShowStringEscapedY	yshow	178
Sine	sin	179
SquareRoot	sqrt	180
StoreVector	astore	181
StringWidth	stringwidth	192
StrokePath	stroke	41
Subtract	sub	50
TransformFont	makefont	182
Translate	translate	12
TranslateT	translateT	188
True	true	44
Truncate	trunc	183
Type	objtype	184
VectorLoad	aload	185
Xor	xor	186

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642874 541

Annex B
(normative)

ASN.1 Object Identifiers and SGML Formal Public Identifiers defined by this International Standard

B.1 Object Identifiers and SGML Formal Public Identifiers defined by this International Standard.

This Normative Annex specifies those Public Object Identifier values which are defined by this International Standard for the identification of information objects.

For each information object which is assigned a Public Object Identifier value, the defining text specifies an Object Name. For each such Object Name, this Annex specifies

-- a public identifier as defined in **ISO/IEC 9070**

— an object identifier value as defined in ISO/IEC 8824

which identify the information object.

B.1.1 Public identifiers

The canonical character string form of the public identifier which identifies an information object to which this International Standard assigns the Object Name *object-name* shall consist of the registered owner prefix

ISO/IEC 10180

followed by "//" and the character sequence *object-name.*

Each public identifier has a semantically equivalent SGML Formal Public Identifier. SGML Formal Public Identifiers are defined in ISO/IEC 8879.

B.1.2 ASN.1 Object Identifier Values

The value reference which identifies the ASN.1 object identifier value which corresponds to each Object Name is listed below The object identifier values which correspond to these value references are specified in 38.1.8.

B.1.2.1 SPDL Identifier

Table B.1

Object Name	Object Identifier Value Reference
SPDL::ExternalID	spdl-exteral-id

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642875 488

B.1.2.2 SPDL Content Identifiers

Table B.2

Object Name	Object Identifier Value Reference
Content::SPDL-Binary	content-spdl-binary
Content::SPSL-ClearText	content-spdl-clear-text

B.1.2.3 Hints

Table B.3

Object Name	Object Identifier Value Reference
Hints::BoundingBox	hints-bounding-box
Hints::NumberOfPages	hints-number-of-pages
Hints::Orientation	hints-orientation
Hints::FontReferences	hints-font-references
Hints::ExternalReferences	hints-external-references
Hints::ResourcesUsed	hints-resources-used
Hints::ColorSpaceFamiliesUsed	hints-color-space-families
Hints::MediaUsed	hints-media-used
Hints::ColorantSetsUsed	hints-colorant-sets-used

B.1.2.4 Color Space Families

Table B.4

Object Name	Object Identifier Value Reference
ColorSpaceFamily::CIELAB	color-space-family-CIELAB
ColorSpaceFamily::CIELUV	color-space-family-CIELUV
ColorSpaceFamily::CIEBasedABC	color-space-family-CIEBasedABC
ColorSpaceFamily::CIEBasedA	color-space-family-CIEBasedA
ColorSpaceFamily::DeviceRGB	color-space-family-DeviceRGB
ColorSpaceFamily::DeviceCMYK	color-space-family-DeviceCMYK
ColorSpaceFamily::DeviceGray	color-space-family-DeviceGray

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642876 314

Object Name	Object Identifier Value Reference
ColorSpaceFamily::Indexed	color-space-family-Indexed
ColorSpaceFamily::NamedColor	color-space-family-NamedColor

B.1.2.5 Color Spaces

Table B.5

Object Name	Object Identifier Value Reference
ColorSpace::DeviceRGB	color-space-DeviceRGB
ColorSpace::DeviceCMYK	color-space-DeviceCMYK
ColorSpace::DeviceKX	color-space-DeviceKX
ColorSpace::DeviceGray	color-space-DeviceGray

B.1.2.6 Data Sources

Table B.6

Object Name	Object Identifier Value Reference
DataSource: Document	data-source-document

B.1.2.7 Filters

Table B.7

Object Name	Object Identifier Value Reference
Filter::ASCIIHexDecode	filter-asciihex-decode
Filter::ASCII85Decode	filter-ascii85-decode
Filter::RunLengthDecode	filter-run-length-decode
Filter::CCITTFaxDecode	filter-ccitt-fax-decode
Filter::NullDecode	filter-null-decode

B.1.2.8 Font Reference Match Rules

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642877 250

Table B.8

Object Name	Object Identifier Value Reference
FontReference::MatchRules::Same	matchrules-same
FontReference::MatchRules::SameIfSpecified	matchrules-sameifspecd

B.1.2.9 Font Reference Satisfaction Criteria

Table B.9

Object Name	Object Identifier Value Reference
FontReference::Satisfaction::Name	satisfaction-name
FontReference::Satisfaction::NameOrProps	satisfaction-nameorprops
FontReference::Satisfaction::Any	satisfaction-any

B.1.2.10 Font Resources

Table B.10

Object Name	Object Identifier Value Reference
Fonts::ISO-Serif::Regular	fonts-isoserif-regular
Fonts::ISO-Serif::Bold	fonts-isoserif-bold
Fonts::ISO-Serif::Italic	fonts-isoserif-italic
Fonts::ISO-Serif::BoldItalic	fonts-isoserif-bolditalic
Fonts::ISO-SanSerif::Regular	fonts-isosanserif-regular
Fonts::ISO-SanSerif::Bold	fonts-isosanserif-bold
Fonts::ISO-SanSerif::Italic	fonts-isosanserif-italic
Fonts::ISO-SanSerif::BoldItalic	fonts-isosanserif-bolditalic
Fonts::ISO-Monospace::Regular	fonts-isomonospace-regular
Fonts::ISO-Monospace::Bold	fonts-isomonospace-bold
Fonts::ISO-Monospace::Italic	fonts-isomonospace-italic
Fonts::ISO-Monospace::BoldItalic	fonts-isomonospace-bolditalic

B.1.2.11 Glyph Index Maps

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642878 197

Table B.11

Object Name	Object Identifier Value Reference
GlyphIndexMap::Latin1Publishing	glyph-index- maps-latin-one
GlyphIndexMap::Latin1PublishingA	glyph-index- maps-latin-one-a
GlyphIndexMap::IR::nnnn	{ glyph-index-maps-ir *n* }

where "*nnnn*" is "0" or the sequence of decimal digits beginning with a non-zero digit which represent the value *n*.

B.1.2.12 Colorant Set Class Document Production Instruction Values

Table B.12

Object Name	Object Identifier Value Reference
dpi::colorant-set-class::cmy	colorant-set-class-cmy
dpi::colorant-set-class::cmyk	colorant-set-class-cmyk
dpi::colorant-set-class::rgb	colorant-set-class-rgb
dpi::colorant-set-class::kx	colorant-set-class-kx
dpi::colorant-set-class::monochrome	colorant-set-class- monochrome

B.1.2.13 Document Abort Policy Document Production Instruction Values

Table B.13

Object Name	Object Identifier Value Reference
dpi::abort-policy::on-warning	dpi-abort-policy-onwarning
dpi::abort-policy::on-error	dpi-abort-policy-onerror
dpi::abort-policy::struggle-on	dpi-abort-policy-struggleon

B.1.3 Object Identifier Value Declarations Module

```
ISO10180-SPDL-OID  {iso(1) standard(0) 10180 modules(1) oid(6)}
DEFINITIONS                                    ::= BEGIN
-- Exports everything --
 -- Imports nothing --
Object-ID                                      ::= OBJECT IDENTIFIER
-- Object Identifier Values --
spdl-id                          Object-ID   ::= { iso(1) standard(0) 10180 }
-- SPDL External Identifier Object Identifier Value --
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642879 023


```
spdl-external-id                Object-ID  ::= { spdl-id  0 }
-- Object Identifier Value Classes --
spdl-modules                    Object-ID  ::= { spdl-id  1 }
spdl-content-types              Object-ID  ::= { spdl-id  2 }
spdl-hints                      Object-ID  ::= { spdl-id  3 }
spdl-color-space-families       Object-ID  ::= { spdl-id  4 }
spdl-color-spaces               Object-ID  ::= { spdl-id  5 }
spdl-fontrefs                   Object-ID  ::= { spdl-id  6 }
spdl-fonts                      Object-ID  ::= { spdl-id  7 }
spdl-glyph-index-maps           Object-ID  ::= { spdl-id  8 }
spdl-dpi                        Object-ID  ::= { spdl-id  9 }
spdl-data-sources               Object-ID  ::= { spdl-id  10 }
spdl-filters                    Object-ID  ::= { spdl-id  11 }
-- Content Type Object Identifier Values --
content-spdl-binary             Object-ID  ::= { spdl-content-types  0 }
content-spdl-clear-text         Object-ID  ::= { spdl-content-types  1 }
-- Hint Object Identifier Values --
hints-bounding-box              Object-ID  ::= { spdl-hints  0 }
hints-number-of pages           Object-ID  ::= { spdl-hints  1 }
hints-orientation               Object-ID  ::= { spdl-hints  2 }
hints-font-references           Object-ID  ::= { spdl-hints  3 }
hints-external-references       Object-ID  ::= { spdl-hints  4 }
hints-resources-used            Object-ID  ::= { spdl-hints  5 }
hints-color-space-families      Object-ID  ::= { spdl-hints  6 }
hints-media-used                Object-ID  ::= { spdl-hints  7 }
hints-colorant-sets-used        Object-ID  ::= { spdl-hints  8 }
-- Color Space Family Object Identifier Values --
color-space-family-CIELAB       Object-ID  ::= { spdl-color-space-families  0 }
color-space-family-CIELUV       Object-ID  ::= { spdl-color-space-families  1 }
color-space-family-CIEBasedABC  Object-ID  ::= {spdl-color-space-families   2 }
color-space-family-CIEBasedA    Object-ID  ::= { spdl-color-space-families  3 }
color-space-family-DeviceRGB    Object-ID  ::= { spdl-color-space-families  4 }
color-space-family-DeviceCMYK   Object-ID  ::= { spdl-color-space-families  5 }
color-space-family-DeviceKX     Object-ID  ::= { spdl-color-space-families  6 }
color-space-family-DeviceGray   Object-ID  ::= { spdl-color-space-families  7 }
color-space-family-Indexed      Object-ID  ::= { spdl-color-space-families  8 }
color-space-family-NamedColor   Object-ID  ::= { spdl-color-space-families  9 }
-- Color Space Object Identifier Values --
color-space-DeviceRGB           Object-ID  ::= { spdl-color-spaces  4 }
color-space-DeviceCMYK          Object-ID  ::= { spdl-color-spaces  5 }
color-space-DeviceKX            Object-ID  ::= { spdl-color-spaces  6 }
color-space-DeviceGray          Object-ID  ::= { spdl-color-spaces  7 }
-- Font Reference Match Rules Object Identifier Values --
fontref-matchrules              Object-ID  ::= { spdl-fontrefs  0 }
matchrules-same                 Object-ID  ::= { fontref-matchrules  0 }
matchrules-sameifspecd          Object-ID  ::= { fontref-matchrules  1 }
-- Font Reference Satisfaction Criterion Object Identifier Values --
fontref-satisfaction            Object-ID  ::= { spdl-fontrefs  1 }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642880 845


```
satisfaction-nameorprops        Object-ID  ::= { fontref-satisfaction  0 }
satisfaction-name               Object-ID  ::= { fontref-satisfaction  1 }
satisfaction-any                Object-ID  ::= { fontref-satisfaction  2 }
-- Font Resource Object Identifier Values --
fonts-isoserif-regular          Object-ID  ::= { spdl-fonts  0 }
fonts-isoserif-bold             Object-ID  ::= { spdl-fonts  1 }
fonts-isoserif-italic           Object-ID  ::= { spdl-fonts  2 }
fonts-isoserif-bolditalic       Object-ID  ::= { spdl-fonts  3 }
fonts-isosanserif-regular       Object-ID  ::= { spdl-fonts  4 }
fonts-isosanserif-bold          Object-ID  ::= { spdl-fonts  5 }
fonts-isosanserif-italic        Object-ID  ::= { spdl-fonts  6 }
fonts-isosanserif-bolditalic    Object-ID  ::= { spdl-fonts  7 }
fonts-isomonospace-regular      Object-ID  ::= { spdl-fonts  8 }
fonts-isomonospace-bold         Object-ID  ::= { spdl-fonts  9 }
fonts-isomonospace-italic       Object-ID  ::= { spdl-fonts  10 }
fonts-isomonospace-bolditalic   Object-ID  ::= { spdl-fonts  11 }
-- Glyph Index Map Object Identifier Values --
glyph-index-maps-ir             Object-ID  ::= { spdl-glyph-index-maps  0 }
glyph-index-maps-latin-one      Object-ID  ::= { spdl-glyph-index-maps  1 }
glyph-index-maps-latin-one-a    Object-ID  ::= { spdl-glyph-index-maps  2 }
-- DPI Object Identifier Values --
-- Colorant Set Class Object Identifier Values --
dpi-colorant-set-classes        Object-ID  ::= { spdl-dpi  0 }
colorant-set-class-cmy          Object-ID  ::= { dpi-colorant-set-classes  0 }
colorant-set-class-cmyk         Object-ID  ::= { dpi-colorant-set-classes  1 }
colorant-set-class-rgb          Object-ID  ::= { dpi-colorant-set-classes  2 }
colorant-set-class-kx           Object-ID  ::= { dpi-colorant-set-class  3 }
colorant-set-class-monochrome   Object-ID  ::= { dpi-colorant-set-class  4 }
-- Abort Policy Object Identifier Values --
dpi-abort-policy                Object-ID  ::= { spdl-dpi  1 }
dpi-abort-policy-onwarning      Object-ID  ::= { dpi-abortpolicy  0 }
dpi-abort-policy-onerror        Object-ID  ::= { dpi-abortpolicy  1 }
dpi-abort-policy-struggleon     Object-ID  ::= { dpi-abortpolicy  2 }
-- Data Source Object Identifier Values --
data-source-document            Object-ID  ::= { spdl-data-sources  0 }
-- Filter Object Identifier Values --
filter-null-decode              Object-ID  ::= { spdl-filters  0 }
filter-asciihex-decode          Object-ID  ::= { spdl-filters  1 }
filter-ascii85-decode           Object-ID  ::= { spdl-filters  2 }
filter-run-length-decode        Object-ID  ::= { spdl-filters  3 }
filter-ccitt-fax-decode         Object-ID  ::= { spdl-filters  4 }
END
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642881 781

© ISO/IEC

Annex C
(normative)

SPDL Indexed Font Representation Format

C.1 Introduction

This normative annex provides the representation for the representation of SPDL Indexed Fonts in SPDL documents.

C.2 Conventions

All encoding elements described in this annex are required except as otherwise stated by use of the term OPTIONAL in parentheses.

The phrases *for public interchange* and *for private interchange* throughout this document refer to the interchange of SPDL Indexed Font information within SPDL documents. This usage should not be confused with the interchange of Font Resources as specified in ISO/IEC 9541.

Throughout this annex, the following typographical emphasis conventions apply:

— Items with **bold** emphasis are literal octet strings which must appear in the Indexed Font encoding exactly as they appear in this annex.

— Items with *italic* emphasis are placeholders for octet strings whose content will be specific to a given Indexed Font.

— Plain text paragraphs contain explanations of encoding elements and outline any restrictions on their content.

There are many instances in this Annex where any one of a small set of literal octet strings is allowed. In the interests of brevity and readability, several placeholders have been defined to represent any of the allowed literal octet strings associated with those placeholders:

The italicized word *rdef* in an encoding element is a placeholder that may be represented in the actual encoding by any one of the following:

def
executeonly def
readonly def

The italicized word *ndef* in an encoding element is a placeholder that may be represented in the actual encoding by any one of the following:

def
executeonly def
readonly def
noaccess def

The italicized word *nput* in an encoding element is a placeholder that may be represented in the actual encoding by any one of the following:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642882 618

put
executeonly put
readonly put
noaccess put

C.3 Definitions

Definitions of terms defined in ISO/IEC 9541 are incorporated herein by reference. Definitions of terms defined in clause 3 of this International Standard 10180 are incorporated by reference.

C.4 SPDL Indexed Font Representation

The SPDL Indexed Font Encoding consists of three sections in sequence: Header, PerFont Attributes, and Glyph Shape Information Each of these sections is described below.

C.4.1 Header

The encoding of an SPDL Indexed Font must begin with the following:

%!*FONTSTANDARD-STANDARDVERSION*: *TYPEFACE DATAVERSION*

where *FONTSTANDARD* is ISO-9541-3 for public interchange, and may be any octet string for private interchange *STANDARDVERSION*, *TYPEFACE*, and DATAVERSION are as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

Following this first required encoding element should be the lines:

%%**/CreationDate**: *TIMESTAMP*

%%*PROPDATA*

where *TIMESTAMP* is an ISO 8824 Universal Time String for public interchange, and may be any octet string for private interchange; *PROPDATA* is as defined in ISO/IEC 9541.

The final encoding element in the header is one or more lines of the form:

% *DATACOPYRIGHT*

and/or

% *DSNCOPYRIGHT*

where *DATACOPYRIGHT* and *DSNCOPYRIGHT* are as defined in ISO/IEC 9541.

C.4.2 PerFont Attributes

The Header of the Indexed Font encoding is immediately followed by the PerFont Attributes. These attributes consist of several top- level attributes and a subset of FontInfo attributes. The top- level attributes and the FontInfo attribute subset in this clause may appear in any order within the PerFont Attributes section of the encoding. The end of the Header and consequent beginning of the PerFont Attributes section is delimited by the line:

FONTDICTSIZE **dict begin**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642883 554

where *FONTDICTSIZE* is a Cardinal whose value is one more than the number of PerFont Attributes actually present in the Indexed Font. For the purpose of computing *FONTDICTSIZE*, the FontInfo attribute subset is considered to be a single attribute.

The end of the PerFont Attributes section is delimited by the line:

currentdict end

All of the PerFont Attributes (including the FontInfo attribute subset), if present in the encoding, must appear between these two delimiter lines. The following subclauses detail the encoding of each top-level attribute, the FontInfo attribute subset, and each attribute within the FontInfo attribute subset.

C.4.2.1 FontInfo (OPTIONAL)

The FontInfo attribute subset contains several attributes, all of which are optional, and which may appear in any order between the FontInfo begin delimiter line:

/FontInfo *NUMFIATTRIBUTES* **dict dup begin**

where *NUMFIATTRIBUTES* is the number of attributes actually present in the FontInfo attribute subset, and the FontInfo end delimiter line:

end *rdef*

The attributes which may be present in the FontInfo attribute subset are encoded as follows:

C.4.2.1.1 Version (OPTIONAL)

/version *(DATAVERSION) rdef*

where *DATAVERSION* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

C.4.2.1.2 Fullname (OPTIONAL)

/FullName *(TYPEFACENAME) rdef*

where *TYPEFACENAME* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

C.4.2.1.3 FamilyName (OPTIONAL)

/FamilyName *(FONTFAMILY) rdef*

where *FONTFAMILY* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

C.4.2.1.4 Weight (OPTIONAL)

/Weight *(WEIGHT) rdef*

where *WEIGHT* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642884 490

C.4.2.1.5 ItalicAngle (OPTIONAL)

/ItalicAngle *POSTUREANGLE rdef*

where *POSTUREANGLE* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

C.4.2.1.6 isFixedPitch (OPTIONAL)

/isFixedPitch *ESCCLASS rdef*

where *ESCCLASS* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange

C.4.2.1.7 UnderlinePosition (OPTIONAL)

/UnderlinePosition *SCOREOFFSETX rdef*

where *SCOREOFFSETX* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

C.4.2.1.8 UnderlineThickness (OPTIONAL)

/UnderlineThickness *SCORETHICK rdef*

where *SCORETHICK* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange

C.4.2.2 FontName (OPTIONAL)

/FontName */FONTNAME rdef*

where *FONTNAME* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange

C.4.2.3 PaintType

/PaintType *STRUCTURE rdef*

where *STRUCTURE* is as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

C.4.2.4 FontType

/FontType *FONTTYPE rdef*

where *FONTTYPE* is of type Cardinal. If the value of the *GlyphShapeTechnology* property (see ISO/IEC 9541-3) is **ISO/STANDARD/9541-3/TYPE1**, the value of *FONTTYPE* is 1. For private interchange, *FONTTYPE* may be any value of type Cardinal between 101 and 998 inclusive

C.4.2.5 FontMatrix

/FontMatrix [*a b c d e f*] *rdef*

or

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

/FontMatrix *{a b c d e f} rdef*

A coordinate (X,Y) in the Glyph Coordinate System (as defined in ISO/IEC 9541) is transformed to a coordinate (x,y) in the SPDL User Coordinate System according to the following linear equations:

$$x = aX + cY + e$$

$$y = bX + dY + f$$

Typically the values of b, c, e, and f are 0, and the values of a and d are both 1/*RELUNITS*, where *RELUNITS* is as defined in ISO/IEC 9541 and as discussed in ISO/IEC 9541-3

C.4.2.6 Encoding

The Encoding attribute may take two forms. The first of these forms is:

/Encoding *PRIVATEENCODING rdef*

where *PRIVATEENCODING* may be any octet string. This form of the Encoding attribute may only be used in private interchange.

The second allowed encoding of the Encoding attribute is a sequence of index-to-glyph-name mappings whose begin is delimited by:

/Encoding 256 array 0 1 255 {1 index exch */DEFAULTGNAME* **put} for**

where *DEFAULTGNAME* is the name to which indices not explicitly mapped in this Encoding attribute are to be mapped.

Following the above delimiter is a sequence of 0 to 256 elements of the form:

dup *INDEX /GNAME* **put**

where *INDEX* is a Cardinal (value between 0 and 255 inclusive) representing the index being mapped to the glyph name; *GNAME* is a glyph identifier as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange.

The sequence of mapping elements is terminated by:

rdef

C.4.2.7 FontBBox

/FontBBox *{MINX MINY MAXX MAXY} rdef*

or

/FontBBox *[MINX MINY MAXX MAXY] rdef*

where *MINX*, *MINY*, *MAXX*, and *MAXY* are as defined under the *MAXFONTEXT* property in ISO/IEC 9541

C.4.2.8 UniqueID (OPTIONAL)

/UniqueID *UNIQUEID rdef*

where *UNIQUEID* is of type Integer. If present, its value should be the same as the value of the UniqueID Private Attribute in the Glyph Shape Information section.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642886 263

C.4.3 Glyph Shape Information

Following the PerFont Attributes section is the Glyph Shape Information section. The entire Glyph Shape Information section is usually encrypted using the **eexec** encryption algorithm (see 3.3.3 of this Annex). If and only if this section is encrypted using the **eexec** algorithm, the encrypted data should be preceded by the delimiter:

currentfile eexec

The actual shape information (and the eexec-encrypted portion of the encoding if **eexec** encryption is used) itself consists of two sections: Private Attributes and CharStrings. The encoding for these two sections before any **eexec** encryption is applied is described in 3.3.1 and 3.3.2; the optional **eexec** encryption algorithm is described in 3.3.3.

C.4.3.1 Private Attributes

The Private Attributes section consists of several required and optional attributes. The semantics of these attributes are described in ISO/IEC 9541-3. These attributes may appear in any order within the Private Attributes section, with the following two exceptions:

— The RD, ND, and NP attributes, or any subset thereof, may appear in any order either before or within the Private Attributes section. The placement of any of these attributes before the Private Attributes section, while allowed, is not recommended.

— The Subrs attribute must be the last attribute.

The beginning of the Private Attributes section is delimited by the following.

dup /Private *PRIVATESIZE* **dict dup begin**

where *PRIVATESIZE* is a Cardinal indicating the number of attributes present in the Private Attributes section, including any RD, ND, or NP attributes which may appear before the Private Attributes section. The end of the Private Attributes section is delimited by the line:

ND

where *ND* is the octet string defined as part of the ND attribute (see 3.3.1.2).

All of the Private Attributes, if present in the encoding, must appear between these two delimiter lines (except RD, ND, and NP as noted above) The following subclauses detail the encoding of each attribute.

C.4.3.1.1 RD

/RD **{string currentfile exch readstring pop}** *rdef*

where *RD* is any octet string not equivalent to the octet string defined as part of the ND attribute, not equivalent to the octet string defined as part of the NP attribute, and not equivalent to the name of any other Private Attribute.

C.4.3.1.2 ND

/ND {ndef} rdef

where *ND* is any octet string not equivalent to the octet string defined as part of the RD attribute, not equivalent to the octet string defined as part of the NP attribute, and not equivalent to the name of any other Private Attribute.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.4.3.1.3 NP

/NP {nput} rdef

where *NP* is any octet string not equivalent to the octet string defined as part of the RD attribute, not equivalent to the octet string defined as part of the ND attribute, and not equivalent to the name of any other Private Attribute.

C.4.3.1.4 BlueValues

/BlueValues [*BLUEVALUEPAIRS*] *ndef*

where *BLUEVALUEPAIRS* is a sequence of 0 to 7 pairs of Integers The semantics of these integer pairs are defined in ISO/IEC 9541-3

C.4.3.1.5 OtherBlues (OPTIONAL)

/OtherBlues [*OTHERBLUESPAIRS*] *ndef*

where *OTHERBLUESPAIRS* is a sequence of 0 to 5 pairs of Integers The semantics of these integer pairs are defined in ISO/IEC 9541-3.

C.4.3.1.6 FamilyBlues (OPTIONAL)

/FamilyBlues [*FAMILYBLUESPAIRS*] *ndef*

where *FAMILYBLUESPAIRS* is a sequence of 0 to 7 pairs of Integers. The semantics of these integer pairs are defined in ISO/IEC 9541-3.

C.4.3.1.7 FamilyOtherBlues (OPTIONAL)

/FamilyOtherBlues [*FAMILYOTHERBLUESPAIRS*] *ndef*

where *FAMILYOTHERBLUESPAIRS* is a sequence of 0 to 5 pairs of Integers. The semantics of these integer pairs are defined in ISO/IEC 9541-3.

C.4.3.1.8 BlueScale (OPTIONAL)

/BlueScale *BLUESCALE ndef*

where *BLUESCALE* is of type Number. The semantics of *BLUESCALE* are defined in ISO/IEC 9541-3.

C.4.3.1.9 BlueShift (OPTIONAL)

/BlueShift *BLUESHIFT ndef*

where *BLUESHIFT* is of type Integer. The semantics of *BLUESHIFT* are defined in ISO/IEC 9541-3.

C.4.3.1.10 BlueFuzz (OPTIONAL)

/BlueFuzz *BLUEFUZZ ndef*

where *BLUEFUZZ* is of type Integer. The semantics of *BLUEFUZZ* are defined in ISO/IEC 9541-3.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642888 036

C.4.3.1.11 StdHW (OPTIONAL)

/StdHW [*STDHW*] *ndef*

where *STDHW* is of type Number. The semantics of *STDHW* are defined in ISO/IEC 9541-3.

C.4.3.1.12 StdVW (OPTIONAL)

/StdVW [*STDVW*] *ndef*

where *STDVW* is of type Number. The semantics of *STDVW* are defined in ISO/IEC 9541-3.

C.4.3.1.13 StemSnapH (OPTIONAL)

/StemSnapH [*STEMSNAPHVALUES*] *ndef*

where *STEMSNAPHVALUES* is a sequence of 0 to 12 Numbers. The semantics of *STEMSNAPHVALUES* are defined in ISO/IEC 9541-3.

C.4.3.1.14 StemSnapV (OPTIONAL)

/StemSnapV [*STEMSNAPVVALUES*] *ndef*

where *STEMSNAPVVALUES* is a sequence of 0 to 12 Numbers. The semantics of *STEMSNAPVVALUES* are defined in ISO/IEC 9541-3

C.4.3.1.15 ForceBold (OPTIONAL)

/ForceBold *FORCEBOLD ndef*

where *FORCEBOLD* is of type Boolean; the value of *FORCEBOLD* must be either **true** or **false**. The semantics of *FORCEBOLD* are defined in ISO/IEC 9541-3.

C.4.3.1.16 LanguageGroup (OPTIONAL)

/LanguageGroup *LANGUAGEGROUP ndef*

where *LANGUAGEGROUP* is of type Cardinal. The semantics of *LANGUAGEGROUP* are defined in ISO/IEC 9541-3.

C.4.3.1.17 lenIV (OPTIONAL)

/lenIV *LENIV ndef*

where *LENIV* is of type Cardinal. The semantics of *LENIV* are defined in ISO/IEC 9541-3.

C.4.3.1.18 MinFeature

/MinFeature {16 16} def

C.4.3.1.19 password

/password 5839 def

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.4.3.1.20 UniqueID (OPTIONAL)

/UniqueID *UNIQUEID rdefn*

where *UNIQUEID* is of type Integer. If present, its value should be the same as the value of the UniqueID PerFont Attribute.

C.4.3.1.21 Subrs

The Subrs attribute consists of a sequence of delimited charstring-encrypted Subrs. The definition of, semantics of, and encryption of Subrs are defined in ISO/IEC 9541-3. The beginning of the Subrs attribute is delimited by:

/Subrs *SUBRSSIZE* **array**

where *SUBRSSIZE* is a Cardinal indicating the number of Subrs in the Subrs attribute.

Following the above delimiter is a sequence of exactly *SUBRSSIZE* elements of the form:

dup *INDEX SUBRLENGTH RD ENCRYPTEDSUBR NP*

where:

— *INDEX* is a unique index of type Cardinal, value between 0 and (*SUBRSSIZE* - 1) inclusive

— *SUBRLENGTH* is the length in octets of the charstring-encrypted Subr

— *RD* is the octet string defined as part of the *RD* attribute

— there is exactly one space character in the encoding between the value of *RD* and the first octet of *ENCRYPTEDSUBR*

— *ENCRYPTEDSUBR* is a charstring-encrypted Subr of length *SUBRLENGTH* octets

— *NP* is the octet string defined as part of the *NP* attribute

The sequence of Subr elements that make up the Subrs attribute is implicitly terminated by the *ND* denoting the end of Private Attributes subsection of the Glyph Shape Information section.

C.4.3.2 CharStrings

The CharStrings subsection of the Glyph Shape Information section consists of a sequence of glyph-name/charstring-encrypted-CharString pairs. The definition of, semantics of, and charstring-encryption of CharStrings are defined in ISO/IEC 9541-3. The beginning of the CharStrings attribute is delimited by:

2 index /CharStrings *CHARSTRINGSSIZE* **dict dup begin**

where *CHARSTRINGSSIZE* is a Cardinal indicating the number of mappings in the CharStrings attribute.

Following the above delimiter is a sequence of exactly *CHARSTRINGSSIZE* elements of the form:

/GNAME CHARSTRINGLENGTH RD ENCRYPTEDCHARSTRING ND

where:

— *GNAME* is a glyph identifier as defined in ISO/IEC 9541 for public interchange, and may be any octet string for private interchange

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642890 794

— *CHARSTRINGLENGTH* is the length in octets of the charstring- encrypted CharString

— *RD* is the octet string defined as part of the *RD* attribute

— there is exactly one space character in the encoding between the value of *RD* and the first octet of *ENCRYPTEDCHARSTRING*

— *ENCRYPTEDCHARSTRING* is a charstring-encrypted CharString of length *CHARSTRINGLENGTH* octets

— *ND* is the octet string defined as part of the *ND* attribute

The end of the sequence of CharString elements (and the CharStrings subsection of the Glyph Shape Information section) is delimited by:

end

end

nput

nput

dup FontName get exch definefont pop

If the Glyph Shape Information section is not to be encrypted using the **eexec** algorithm, the above also denotes the end of the Glyph Shape Information section and the end of the entire encoding.

If and only if the Glyph Shape Information section is to be encrypted using the **eexec** algorithm, the following should appear as the final octet string inside the **eexec**-encrypted Glyph Shape Information section:

mark currentfile closefile

and the following should appear as cleartext (non-**eexec**-e ncrypted) after the **eexec**-encrypted Glyph Shape Information Section to delimit the end of the entire encoding:

cleartomark

C.4.3.3 eexec Encryption Algorithm

The **eexec** encryption algorithm is used to encrypt the private Attributes and CharStrings and a nearly identical algorithm is used to decrypt these sections of the SPDL Indexed Font representation.

C.4.3.3.1 Encryption

To encrypt a sequence of plaintext octets using the **eexec** encryption algorithm:

a) Generate 4 random numbers between 0 and 255 inclusive and form an octet string of length 4 from these numbers. Place this octet string immediately before the octet string which is the plain text.

b) Initialize an unsigned 16 bit integer variable R to 55,665.

c) For each octet, Q, in the octet string formed in step 1. execute the following steps, replacing Q with its encrypted form E:

   1) Assign the high order 8 bits of R to a temporary variable, T.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642891 620

2) Exclusive-or Q with T, producing a encrypted octet, E.

3) Compute the next value of R using the formula

4) R = ((E + R) 4 52845 + 22719) modulo 65,536

C.4.3.3.2 Decryption

To decrypt a sequence of **eexec** encrypted octets using the **eexec** decryption algorithm:

a) Initialize an unsigned 16 bit integer variable R to 55,665.

b) For each octet, E, in the encrypted octet string, execute the following steps, replacing E with its decrypted form Q:

1) Assign the high order 8 bits of R to a temporary variable, T.

2) Exclusive-or E with T, producing a decrypted octet, Q.

3) Compute the next value of R using the formula

4) R = ((E + R) 4 52845 + 22719) modulo 65,536

c) Discard the first 4 octets of the resulting decrpyted octet string. The remaining octets are the decrypted text.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642892 567

Annex D
(normative)

SPDL mandatory font set for interchange

All conforming SPDL implementations shall support a minimum standard font set for interchange consisting of a Regular, Bold, Italic, and BoldItalic face for each of three families: *iso-serif, iso-sanserif,* and *iso-monospace.* These three families shall be fonts which consist of typographically-spaced serifed glyphs, typographically-spaced glyphs without serifs, and monospaced glyphs with serifs, respectively. This Annex describes the required glyph complement and metrics for each of these fonts. Implementations may use any fonts of appropriate design and style as described above to implement the required glyph complement, and shall then impose the metrics given in this Annex

Table D.1 provides a standard glyph complement and width information for the SPDL standard font set for interchange consisting of a Regular, Bold, Italic, and BoldItalic face for each of three families: *iso-serif, iso-sanserif,* and *iso-monospace.* Each table entry consists of an AFII Glyph ID (informative) and Glyph Description taken from the January 1990 AFII-Registry/LatinPublishing (Association for Font Information Interchange Glyph Collection for LatinPublishing, Western) glyph collection, and widths for that glyph for each of the *iso-serif* and *iso-sanserif* faces, assuming a glyph representation associated with a font resource with an ISO-9541/RELUNITS value of 1000 The faces in the *iso-monospace* family (*iso-monospaceRegular, iso-monospaceBold, iso-monospaceItalic,*and *iso-monospaceBoldItalic*) also have the glyph complement listed in Table D.1, and the width of each glyph in each face is 600

Table D.1

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
000/040	Space (Normally non-printing)	250	250	250	250	278	278	278	278
000/041	Exclamation mark	333	333	333	389	278	333	278	333
000/042	Quote, Double, Neutral	408	555	420	555	355	474	355	474
000/043	Number sign	500	500	500	500	556	556	556	556
000/044	General currency symbol	500	500	500	500	556	556	556	556
000/045	Percent sign	833	1000	833	833	889	889	889	889
000/046	Ampersand	778	833	778	778	667	722	667	722
000/050	Parenthesis, beginning (open), curved	333	333	333	333	333	333	333	333
000/051	Parenthesis, close (end), curved	333	333	333	333	333	333	333	333
000/052	Asterisk	500	500	500	500	389	389	389	389
000/053	Plus sign	564	570	675	570	584	584	584	584

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642893 4T3

Table D.1 *(continued)*

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
000/054	Comma	250	250	250	250	278	278	278	278
000/056	Period	250	250	250	250	278	278	278	278
000/057	Slant	278	278	278	278	278	278	278	278
000/060	Zero	500	500	500	500	556	556	556	556
000/061	One	500	500	500	500	556	556	556	556
000/062	Two	500	500	500	500	556	556	556	556
000/063	Three	500	500	500	500	556	556	556	556
000/064	Four	500	500	500	500	556	556	556	556
000/065	Five	500	500	500	500	556	556	556	556
000/066	Six	500	500	500	500	556	556	556	556
000/067	Seven	500	500	500	500	556	556	556	556
000/070	Eight	500	500	500	500	556	556	556	556
000/071	Nine	500	500	500	500	556	556	556	556
000/072	Colon	278	333	333	333	278	333	278	333
000/073	Semicolon	278	333	333	333	278	333	278	333
000/074	Less than	564	570	675	570	584	584	584	584
000/075	Equals	564	570	675	570	584	584	584	584
000/076	Greater than	564	570	675	570	584	584	584	584
000/077	Question mark	444	500	500	500	556	611	556	611
000/100	At sign (Commercial at sign)	921	930	920	832	1015	975	1015	975
000/101	Uppercase Latin letter A	722	722	611	667	667	722	667	722
000/102	Uppercase Latin letter B	667	667	611	667	667	722	667	722
000/103	Uppercase Latin letter C	667	722	667	667	722	722	722	722
000/104	Uppercase Latin letter D	722	722	722	722	722	722	722	722
000/105	Uppercase Latin letter E	611	667	611	667	667	667	667	667
000/106	Uppercase Latin letter F	556	611	611	667	611	611	611	611
000/107	Uppercase Latin letter G	722	778	722	722	778	778	778	778

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table D.1 (continued)

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
000/110	Uppercase Latin letter H	722	778	722	778	722	722	722	722
000/111	Uppercase Latin letter I	333	389	333	389	278	278	278	278
000/112	Uppercase Latin letter J	389	500	444	500	500	556	500	556
000/113	Uppercase Latin letter K	722	778	667	667	667	722	667	722
000/114	Uppercase Latin letter L	611	667	556	611	556	611	556	611
000/115	Uppercase Latin letter M	889	944	833	889	833	833	833	833
000/116	Uppercase Latin letter N	722	722	667	722	722	722	722	722
000/117	Uppercase Latin letter O	722	778	722	722	778	778	778	778
000/120	Uppercase Latin letter P	556	611	611	611	667	667	667	667
000/121	Uppercase Latin letter Q	722	778	722	722	778	778	778	778
000/122	Uppercase Latin letter R	667	722	611	667	722	722	722	722
000/123	Uppercase Latin letter S	556	556	500	556	667	667	667	667
000/124	Uppercase Latin letter T	611	667	556	611	611	611	611	611
000/125	Uppercase Latin letter U	722	722	722	722	722	722	722	722
000/126	Uppercase Latin letter V	722	722	611	667	667	667	667	667
000/127	Uppercase Latin letter W	944	1000	833	889	944	944	944	944
000/130	Uppercase Latin letter X	722	722	611	667	667	667	667	667
000/131	Uppercase Latin letter Y	722	722	556	611	667	667	667	667
000/132	Uppercase Latin letter Z	611	667	556	611	611	611	611	611
000/133	Bracket, Opening (Left)	333	333	389	333	278	333	278	333
000/134	Slant, Reverse	278	278	278	278	278	278	278	278
000/135	Bracket, Closing (Right)	333	333	389	333	278	333	278	333
000/136	Circumflex (l.c. spacing accent)	333	333	333	333	333	333	333	333
000/137	Bar, Low	500	500	500	500	556	556	556	556
000/140	Grave (l.c spacing accent)	333	333	333	333	333	333	333	333
000/141	Lowercase Latin letter a	444	500	500	500	556	556	556	556
000/142	Lowercase Latin letter b	500	556	500	500	556	611	556	611
000/143	Lowercase Latin letter c	444	444	444	444	500	556	500	556

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642895 276

Table D.1 (continued)

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
000/144	Lowercase Latin letter d	500	556	500	500	556	611	556	611
000/145	Lowercase Latin letter e	444	444	444	444	556	556	556	556
000/146	Lowercase Latin letter f	333	333	278	333	278	333	278	333
000/147	Lowercase Latin letter g	500	500	500	500	556	611	556	611
000/150	Lowercase Latin letter h	500	556	500	556	556	611	556	611
000/151	Lowercase Latin letter i	278	278	278	278	222	278	222	278
000/152	Lowercase Latin letter j	278	333	278	278	222	278	222	278
000/153	Lowercase Latin letter k	500	556	444	500	500	556	500	556
000/154	Lowercase Latin letter l	278	278	278	278	222	278	222	278
000/155	Lowercase Latin letter m	778	833	722	778	833	889	833	889
000/156	Lowercase Latin letter n	500	556	500	556	556	611	556	611
000/157	Lowercase Latin letter o	500	500	500	500	556	611	556	611
000/160	Lowercase Latin letter p	500	556	500	500	556	611	556	611
000/161	Lowercase Latin letter q	500	556	500	500	556	611	556	611
000/162	Lowercase Latin letter r	333	444	389	389	333	389	333	389
000/163	Lowercase Latin letter s	389	389	389	389	500	556	500	556
000/164	Lowercase Latin letter t	278	333	278	278	278	333	278	333
000/165	Lowercase Latin letter u	500	556	500	556	556	611	556	611
000/166	Lowercase Latin letter v	500	500	444	444	500	556	500	556
000/167	Lowercase Latin letter w	722	722	667	667	722	778	722	778
000/170	Lowercase Latin letter x	500	500	444	500	500	556	500	556
000/171	Lowercase Latin letter y	500	500	444	444	500	556	500	556
000/172	Lowercase Latin letter z	444	444	389	389	500	500	500	500
000/173	Brace, beginning (open)	480	394	400	348	334	389	334	389
000/174	Bar, vertical, single	200	220	275	220	260	280	260	280
000/175	Brace, closing (end)	480	394	400	348	334	389	334	389
000/176	Tilde (l.c. spacing accent)	333	333	333	333	333	333	333	333
000/241	Exclamation mark (Spanish) Inverted	333	333	389	389	333	333	333	333

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table D.1 (continued)

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
000/242	Cent sign	500	500	500	500	556	556	556	556
000/243	Pound-Sterling sign	500	500	500	500	556	556	556	556
000/244	Dollar sign	500	500	500	500	556	556	556	556
000/245	Yen currency symbol (Japanese)	500	500	500	500	556	556	556	556
000/247	Section sign	500	500	500	500	556	556	556	556
000/251	Quotation mark, beginning, single, curved	333	333	333	333	222	278	222	278
000/252	Quotation mark, beginning, double, curved	444	500	556	500	333	500	333	500
000/253	Guillemet (European quotation), left, double	500	500	500	500	556	556	556	556
000/260	Degree sign	400	400	400	400	400	400	400	400
000/261	Plus or minus sign	564	570	675	570	584	584	584	584
000/262	Squared	300	300	300	300	333	333	333	333
000/263	Cubed	300	300	300	300	333	333	333	333
000/264	Multiply sign	564	570	675	570	584	584	584	584
000/265	Micro sign	500	556	500	576	556	611	556	611
000/266	Paragraph sign	453	540	523	500	537	556	537	556
000/267	Dot, Centered	250	250	250	250	278	278	278	278
000/270	Divide sign	564	570	675	570	584	584	584	584
000/271	Quotation mark, closing, single, curved	333	333	333	333	222	278	222	278
000/272	Quotation mark, closing, double, curved	444	500	556	500	333	500	333	500
000/273	Guillemet (European quotation), right, double	500	500	500	500	556	556	556	556
000/274	Fraction, one quarter, en set	750	750	750	750	834	834	834	834
000/275	Fraction, one half, en set	750	750	750	750	834	834	834	834
000/276	Fraction, three quarters, en set	750	750	750	750	834	834	834	834

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642897 049

Table D.1 (continued)

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
000/277	Question mark, Spanish, inverted	444	500	500	500	611	611	611	611
000/321	Reference 1	300	300	300	300	333	333	333	333
000/322	Registered sign	760	747	760	747	737	737	737	737
000/323	Copyright sign	760	747	760	747	737	737	737	737
000/324	Trademark sign	980	1000	980	1000	1000	1000	1000	1000
000/341	AE digraph, Uppercase	889	1000	889	944	1000	1000	1000	1000
000/342	D with stroke, Uppercase (Croatian)	722	722	722	722	722	722	722	722
000/343	Ordinal Indicator, Feminine (Spanish)	276	300	276	266	370	370	370	370
000/350	L with stroke, Uppercase (Polish)	611	667	556	611	556	611	556	611
000/351	O with slash, Uppercase (Norwegian, Danish)	722	778	722	722	778	778	778	778
000/352	OE digraph, Uppercase	889	1000	944	944	1000	1000	1000	1000
000/353	Ordinal Indicator, Masculine (Spanish)	310	330	310	300	365	365	365	365
000/354	Thorn, Uppercase (Icelandic)	556	611	611	611	667	667	667	667
000/361	ae digraph, Lowercase	667	722	667	722	889	889	889	889
000/363	Eth, Lowercase (Icelandic)	500	500	500	500	556	611	556	611
000/365	i, dotless, Lowercase (Turkish)	278	278	278	278	278	278	278	278
000/370	l with stroke, Lowercase (Polish)	278	278	278	278	222	278	222	278
000/371	o with slash, Lowercase (Norwegian, Danish)	500	500	500	500	611	611	611	611
000/372	oe digraph, Lowercase	722	722	667	722	944	944	944	944
000/373	s, Double	500	556	500	500	611	611	611	611
000/374	Thorn, Lowercase (Icelandic)	500	556	500	500	556	611	556	611
041/076	Hyphen	333	333	333	333	333	333	333	333

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642898 T85

Table D.1 (continued)

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
041/104	Leader, three-dot on an em body	1000	1000	889	1000	1000	1000	1000	1000
042/300	Up arrowhead	469	581	422	570	469	584	469	584
043/042	Diaeresis (l.c. spacing accent)	333	333	333	333	333	333	333	333
043/043	Acute (l.c. spacing accent)	333	333	333	333	333	333	333	333
043/044	Macron	333	333	333	333	333	333	333	333
043/045	Breve (l.c spacing accent)	333	333	333	333	333	333	333	333
043/046	Over-dot (l.c spacing accent)	333	333	333	333	333	333	333	333
043/050	Over-ring (l.c spacing accent)	333	333	333	333	333	333	333	333
043/051	Double acute (l c spacing accent)	333	333	333	333	333	333	333	333
043/052	Ogonek (l.c. spacing undermark)	333	333	333	333	333	333	333	333
043/053	Hachek (l c. spacing accent)	333	333	333	333	333	333	333	333
043/054	Cedilla (l.c. spacing undermark)	333	333	333	333	333	333	333	333
043/262	Quote, Single. Left, Lowered	333	333	333	333	222	278	222	278
356/055	Minus sign	564	570	675	606	584	584	584	584
356/176	Similar to, Type 1	541	520	541	570	584	584	584	584
357/044	Dash, En	500	500	500	500	556	556	556	556
357/045	Dash, Em	1000	1000	889	1000	1000	1000	1000	1000
357/047	Quote, Single, Neutral	180	278	214	278	191	238	191	238
357/050	Quote, Double, Left, Lowered	444	500	556	500	333	500	333	500
357/052	Guillemet, Single, Left quote	333	333	333	333	333	333	333	333
357/053	Guillemet, Single, Right quote	333	333	333	333	333	333	333	333

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642899 911

© ISO/IEC

Table D.1 (continued)

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
357/060	Dagger	500	500	500	500	556	556	556	556
357/061	Dagger, double	500	500	500	500	556	556	556	556
357/101	Per thousand	1000	1000	1000	1000	1000	1000	1000	1000
357/146	Bullet, centered	350	350	350	350	350	350	350	350
357/152	End of Line Symbol; Also: Not	564	570	675	606	584	584	584	584
357/153	Bar, Vertical, Broken	200	220	275	220	260	280	260	280
357/242	Florin	500	500	500	500	556	556	556	556
360/044	Ligature fi	556	556	500	556	500	611	500	611
360/045	Ligature fl	556	556	500	556	500	611	500	611
361/041	Grave A	722	722	611	667	667	722	667	722
361/042	Acute A	722	722	611	667	667	722	667	722
361/043	Circumflex A	722	722	611	667	667	722	667	722
361/044	Tilde A	722	722	611	667	667	722	667	722
361/047	Diaeresis A	722	722	611	667	667	722	667	722
361/050	Ring A	722	722	611	667	667	722	667	722
361/055	Cedilla C	667	722	667	667	722	722	722	722
361/060	Grave E	611	667	611	667	667	667	667	667
361/061	Acute E	611	667	611	667	667	667	667	667
361/062	Circumflex E	611	667	611	667	667	667	667	667
361/065	Diaeresis E	611	667	611	667	667	667	667	667
361/076	Grave I	333	389	333	389	278	278	278	278
361/077	Acute I	333	389	333	389	278	278	278	278
361/100	Circumflex I	333	389	333	389	278	278	278	278
361/104	Diaeresis I	333	389	333	389	278	278	278	278
361/114	Tilde N	722	722	667	722	722	722	722	722
361/117	Grave O	722	778	722	722	778	778	778	778
361/120	Acute O	722	778	722	722	778	778	778	778
361/121	Circumflex O	722	778	722	722	778	778	778	778

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642900 463

Table D.1 (continued)

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
361/122	Tilde O	722	778	722	722	778	778	778	778
361/124	Diaeresis O	722	778	722	722	778	778	778	778
361/134	Hachek S	556	556	500	556	667	667	667	667
361/137	Grave U	722	722	722	722	722	722	722	722
361/140	Acute U	722	722	722	722	722	722	722	722
361/141	Circumflex U	722	722	722	722	722	722	722	722
361/145	Diaeresis U	722	722	722	722	722	722	722	722
361/153	Acute Y	722	722	556	611	667	667	667	667
361/155	Diaeresis Y	722	722	556	611	667	667	667	667
361/160	Hachek Z	611	667	556	611	611	611	611	611
361/241	Grave a	444	500	500	500	556	556	556	556
361/242	Acute a	444	500	500	500	556	556	556	556
361/243	Circumflex a	444	500	500	500	556	556	556	556
361/244	Tilde a	444	500	500	500	556	556	556	556
361/247	Diaeresis a	444	500	500	500	556	556	556	556
361/250	Ring a	444	500	500	500	556	556	556	556
361/255	Cedilla c	444	444	444	444	500	556	500	556
361/260	Grave e	444	444	444	444	556	556	556	556
361/261	Acute e	444	444	444	444	556	556	556	556
361/262	Circumflex e	444	444	444	444	556	556	556	556
361/265	Diaeresis e	444	444	444	444	556	556	556	556
361/276	Grave i	278	278	278	278	278	278	278	278
361/277	Acute i	278	278	278	278	278	278	278	278
361/300	Circumflex i	278	278	278	278	278	278	278	278
361/304	Diaeresis i	278	278	278	278	278	278	278	278
361/314	Tilde n	500	556	500	556	556	611	556	611
361/317	Grave o	500	500	500	500	556	611	556	611
361/320	Acute o	500	500	500	500	556	611	556	611
361/321	Circumflex o	500	500	500	500	556	611	556	611

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642901 3TT

Table D.1 (concluded)

Informative AFII Glyph ID (IR/octal#)	AFII Glyph Description	Serif Regular	Serif Bold	Serif Italic	Serif Bold Italic	Sans Serif Regular	Sans Serif Bold	Sans Serif Italic	Sans Serif Bold Italic
361/322	Tilde o	500	500	500	500	556	611	556	611
361/324	Diaeresis o	500	500	500	500	556	611	556	611
361/334	Hachek s	389	389	389	389	500	556	500	556
361/337	Grave u	500	556	500	556	556	611	556	611
361/340	Acute u	500	556	500	556	556	611	556	611
361/341	Circumflex u	500	556	500	556	556	611	556	611
361/345	Diaeresis u	500	556	500	556	556	611	556	611
361/353	Acute y	500	500	444	444	500	556	500	556
361/355	Diaeresis y	500	500	444	444	500	556	500	556
361/360	Hachek z	444	444	389	389	500	500	500	500
375/257	Fraction bar	167	167	167	167	167	167	167	167

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642902 236

Annex E
(normative)

Glyph Index Maps

This normative annex defines two Glyph Index Maps -- the mandatory standard *GlyphIndexMap::Latin1Publishing*, and the *GlyphIndexMap::Latin1PublishingA*.

E.1 GlyphIndexMap::Latin1Publishing

Table E.1

Index	Glyph ID	AFII Description (Informative)
0	.notdef	unused
1	.notdef	unused
2	.notdef	unused
3	.notdef	unused
4	.notdef	unused
5	.notdef	unused
6	.notdef	unused
7	.notdef	unused
8	.notdef	unused
9	.notdef	unused
10	.notdef	unused
11	.notdef	unused
12	.notdef	unused
13	.notdef	unused
14	.notdef	unused
15	.notdef	unused
16	.notdef	unused
17	.notdef	unused
18	.notdef	unused
19	.notdef	unused

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642903 172

Table E.1 (continued)

Index	Glyph ID	AFII Description (Informative)
20	.notdef	unused
21	.notdef	unused
22	.notdef	unused
23	.notdef	unused
24	.notdef	unused
25	.notdef	unused
26	.notdef	unused
27	.notdef	unused
28	.notdef	unused
29	.notdef	unused
30	.notdef	unused
31	.notdef	unused
32	afii32	SPACE
33	afii33	EXCLAMATION MARK
34	afii33	QUOTATION MARK
35	afii35	NUMBER SIGN
36	afii36	DOLLAR SIGN
37	afii37	PERCENT SIGN
38	afii38	AMPERSAND
39	afii39	APOSTROPHE
40	afii40	LEFT PARENTHESIS
41	afii41	RIGHT PARENTHESIS
42	afii42	ASTERISK
43	afii43	PLUS SIGN
44	afii44	COMMA
45	afii45	HYPHEN, MINUS SIGN
46	afii46	FULL STOP
47	afii47	SOLIDUS
48	afii48	DIGIT ZERO
49	afii49	DIGIT ONE

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642904 009

Table E.1 (continued)

Index	Glyph ID	AFII Description (Informative)
50	afii50	DIGIT TWO
51	afii51	DIGIT THREE
52	afii52	DIGIT FOUR
53	afii53	DIGIT FIVE
54	afii54	DIGIT SIX
55	afii55	DIGIT SEVEN
56	afii56	DIGIT EIGHT
57	afii57	DIGIT NINE
58	afii58	COLON
59	afii59	SEMICOLON
60	afii60	LESS-THAN SIGN
61	afii61	EQUALS SIGN
62	afii62	GREATER-THAN SIGN
63	afii63	QUESTION MARK
64	afii.64	COMMERCIAL AT
65	afii65	CAPITAL LETTER A
66	afii66	CAPITAL LETTER B
67	afii67	CAPITAL LETTER C
68	afii68	CAPITAL LETTER D
69	afii69	CAPITAL LETTER E
70	afii70	CAPITAL LETTER F
71	afii71	CAPITAL LETTER G
72	afii72	CAPITAL LETTER H
73	afii73	CAPITAL LETTER I
74	afii74	CAPITAL LETTER J
75	afii75	CAPITAL LETTER K
76	afii76	CAPITAL LETTER L
77	afii77	CAPITAL LETTER M
78	afii78	CAPITAL LETTER N
79	afii79	CAPITAL LETTER O

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642905 T45

Table E.1 (continued)

Index	Glyph ID	AFII Description (Informative)
80	afii80	CAPITAL LETTER P
81	afii81	CAPITAL LETTER Q
82	afii82	CAPITAL LETTER R
83	afii83	CAPITAL LETTER S
84	afii84	CAPITAL LETTER T
85	afii85	CAPITAL LETTER U
86	afii86	CAPITAL LETTER V
87	afii87	CAPITAL LETTER W
88	afii88	CAPITAL LETTER X
89	afii89	CAPITAL LETTER Y
90	afii90	CAPITAL LETTER Z
91	afii91	LEFT SQUARE BRACKET
92	afii92	REVERSE SOLIDUS
93	afii93	RIGHT SQUARE BRACKET
94	afii94	CIRCUMFLEX ACCENT
95	afii95	LOW LINE
96	afii96	GRAVE ACCENT
97	afii97	SMALL LETTER a
98	afii98	SMALL LETTER b
99	afii99	SMALL LETTER c
100	afii100	SMALL LETTER d
101	afii101	SMALL LETTER e
102	afii102	SMALL LETTER f
103	afii103	SMALL LETTER g
104	afii104	SMALL LETTER h
105	afii105	SMALL LETTER i
106	afii106	SMALL LETTER j
107	afii107	SMALL LETTER k
108	afii108	SMALL LETTER l
109	afii109	SMALL LETTER m

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table E.1 (continued)

Index	Glyph ID	AFII Description (Informative)
110	afii110	SMALL LETTER n
111	afii111	SMALL LETTER o
112	afii112	SMALL LETTER p
113	afii113	SMALL LETTER q
114	afii114	SMALL LETTER r
115	afii115	SMALL LETTER s
116	afii116	SMALL LETTER t
117	afii117	SMALL LETTER u
118	afii118	SMALL LETTER v
119	afii119	SMALL LETTER w
120	afii120	SMALL LETTER x
121	afii121	SMALL LETTER y
122	afii122	SMALL LETTER z
123	afii123	LEFT CURLY BRACE
124	afii124	VERTICAL LINE
125	afii125	RIGHT CURLY BRACE
126	afii126	TILDE
127	.notdef	unused
128	.notdef	unused
129	.notdef	unused
130	.notdef	unused
131	.notdef	unused
132	.notdef	unused
133	.notdef	unused
134	.notdef	unused
135	.notdef	unused
136	.notdef	unused
137	.notdef	unused
138	.notdef	unused
139	.notdef	unused

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642907 818

Table E.1 *(continued)*

Index	Glyph ID	AFII Description (Informative)
140	.notdef	unused
141	.notdef	unused
142	.notdef	unused
143	.notdef	unused
144	.notdef	unused
145	.notdef	unused
146	.notdef	unused
147	.notdef	unused
148	.notdef	unused
149	.notdef	unused
150	.notdef	unused
151	.notdef	unused
152	.notdef	unused
153	.notdef	unused
154	.notdef	unused
155	.notdef	unused
156	.notdef	unused
157	.notdef	unused
158	.notdef	unused
159	.notdef	unused
160	afii122401	NO-BREAK SPACE
161	afii161	INVERTED EXCLAMATION MARK
162	afii162	CENT SIGN
163	afii163	POUND SIGN
164	afii36	CURRENCY SIGN
165	afii165	YEN SIGN
166	afii122475	BROKEN BAR
167	afii167	PARAGRAPH SIGN, SECTION SIGN
168	afii17954	DIAERESIS
169	afii211	COPYRIGHT SIGN

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table E.1 (continued)

Index	Glyph ID	AFII Description (Informative)
170	afii227	FEMININE ORDINAL INDICATOR
171	afii171	LEFT ANGLE QUOTATION MARK
172	afii122474	NOT SIGN
173	afii122403	SOFT HYPHEN
174	afii210	REGISTERED TRADE MARK SIGN
175	afii17956	MACRON
176	afii17960	RING ABOVE, DEGREE SIGN
177	afii177	PLUS-MINUS SIGN
178	afii178	SUPERSCRIPT TWO
179	afii179	SUPERSCRIPT THREE
180	afii17955	ACUTE ACCENT
181	afii181	MICRO SIGN
182	afii182	PILCROW SIGN
183	afii183	MIDDLE DOT
184	afii17964	CEDILLA
185	afii209	SUPERSCRIPT ONE
186	afii235	MASCULINE ORDINAL INDICATOR
187	afii187	RIGHT ANGLE QUOTATION MARK
188	afii188	VULGAR FRACTION ONE QUARTER
189	afii189	VULGAR FRACTION ONE HALF
190	afii190	VULGAR FRACTION THREE QUARTERS
191	afii191	INVERTED QUESTION MARK
192	afii123425	CAPITAL LETTER A WITH GRAVE ACCENT
193	afii123426	CAPITAL LETTER A WITH ACUTE ACCENT
194	afii123427	CAPITAL LETTER A WITH CIRCUMFLEX ACCENT
195	afii123428	CAPITAL LETTER A WITH TILDE
196	afii123429	CAPITAL LETTER A WITH DIAERESIS
197	afii123430	CAPITAL LETTER A WITH RING ABOVE
198	afii225	CAPITAL DIPTHONG A WITH E
199	afii123437	CAPITAL LETTER C WITH CEDILLA

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642909 690

Table E.1 (continued)

Index	Glyph ID	AFII Description (Informative)
200	afii123440	CAPITAL LETTER E WITH GRAVE ACCENT
201	afii123441	CAPITAL LETTER E WITH ACUTE ACCENT
202	afii123442	CAPITAL LETTER E WITH CIRCUMFLEX ACCENT
203	afii123445	CAPITAL LETTER E WITH DIAERESIS
204	afii123454	CAPITAL LETTER I WITH GRAVE ACCENT
205	afii123455	CAPITAL LETTER I WITH ACUTE ACCENT
206	afii123456	CAPITAL LETTER I WITH CIRCUMFLEX ACCENT
207	afii123460	CAPITAL LETTER I WITH DIAERESIS
208	afii226	CAPITAL ICELANDIC LETTER ETH
209	afii123468	CAPITAL LETTER N WITH TILDE
210	afii123471	CAPITAL LETTER O WITH GRAVE ACCENT
211	afii123472	CAPITAL LETTER O WITH ACUTE ACCENT
212	afii123473	CAPITAL LETTER O WITH CIRCUMFLEX ACCENT
213	afii123474	CAPITAL LETTER O WITH TILDE
214	afii123475	CAPITAL LETTER O WITH DIAERESIS
215	afii180	MULTIPLICATION SIGN
216	afii233	CAPITAL LETTER O WITH OBLIQUE STROKE
217	afii123487	CAPITAL LETTER U WITH GRAVE ACCENT
218	afii123488	CAPITAL LETTER U WITH ACUTE ACCENT
219	afii123489	CAPITAL LETTER U WITH CIRCUMFLEX ACCENT
220	afii123493	CAPITAL LETTER U WITH DIAERESIS
221	afii123499	CAPITAL LETTER Y WITH ACUTE ACCENT
222	afii236	CAPITAL ICELANDIC LETTER THORN
223	afii251	SMALL GERMAN LETTER SHARP s
224	afii123553	SMALL LETTER a WITH GRAVE ACCENT
225	afii123554	SMALL LETTER a WITH ACUTE ACCENT
226	afii123555	SMALL LETTER a WITH CIRCUMFLEX ACCENT
227	afii123556	SMALL LETTER a WITH TILDE
228	afii123559	SMALL LETTER a WITH DIAERESIS
229	afii123560	SMALL LETTER a WITH RING ABOVE

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642910 302

Table E.1 (concluded)

Index	Glyph ID	AFII Description (Informative)
230	afii241	SMALL DIPTHONG a WITH e
231	afii123565	SMALL LETTER c WITH CEDILLA
232	afii123568	SMALL LETTER e WITH GRAVE ACCENT
233	afii123569	SMALL LETTER e WITH ACUTE ACCENT
234	afii123570	SMALL LETTER e WITH CIRCUMFLEX ACCENT
235	afii123573	SMALL LETTER e WITH DIAERESIS
236	afii123582	SMALL LETTER i WITH GRAVE ACCENT
237	afii123583	SMALL LETTER i WITH ACUTE ACCENT
238	afii123584	SMALL LETTER i WITH CIRCUMFLEX ACCENT
239	afii123588	SMALL LETTER i WITH DIAERESIS
240	afii234	SMALL ICELANDIC LETTER ETH
241	afii123596	SMALL LETTER n WITH TILDE
242	afii123599	SMALL LETTER o WITH GRAVE ACCENT
243	afii123600	SMALL LETTER o WITH ACUTE ACCENT
244	afii123601	SMALL LETTER o WITH CIRCUMFLEX ACCENT
245	afii123602	SMALL LETTER o WITH TILDE
246	afii123604	SMALL LETTER o WITH DIAERESIS
247	afii184	DIVISION SIGN
248	afii249	SMALL LETTER o WITH OBLIQUE STROKE
249	afii123615	SMALL LETTER u WITH GRAVE ACCENT
250	afii123616	SMALL LETTER u WITH ACUTE ACCENT
251	afii123617	SMALL LETTER u WITH CIRCUMFLEX ACCENT
252	afii123621	SMALL LETTER u WITH DIAERESIS
253	afii123627	SMALL LETTER y WITH ACUTE ACCENT
254	afii252	SMALL ICELANDIC LETTER THORN
255	afii123629	SMALL LETTER y WITH DIAERESIS

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642911 249

© ISO/IEC

E.2 GlyphIndexMap::Latin1PublishingA

Table E.2

Index	Glyph ID
0	.notdef
1	.notdef
2	.notdef
3	.notdef
4	.notdef
5	.notdef
6	.notdef
7	.notdef
8	.notdef
9	.notdef
10	.notdef
11	.notdef
12	.notdef
13	.notdef
14	.notdef
15	.notdef
16	.notdef
17	.notdef
18	.notdef
19	.notdef
20	.notdef
21	.notdef
22	.notdef
23	.notdef
24	.notdef
25	.notdef
26	.notdef
27	.notdef
28	.notdef

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642912 185

Table E.2 (continued)

Index	Glyph ID
29	.notdef
30	.notdef
31	.notdef
32	space
33	exclam
34	quotedbl
35	numbersign
36	dollar
37	percent
38	ampersand
39	quoteright
40	parenleft
41	parenright
42	asterisk
43	plus
44	comma
45	minus
46	period
47	slash
48	zero
49	one
50	two
51	three
52	four
53	five
54	six
55	seven
56	eight
57	nine
58	colon

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642913 011

Table E.2 (continued)

Index	Glyph ID
59	semicolon
60	less
61	equal
62	greater
63	question
64	at
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table E.2 (continued)

Index	Glyph ID
89	Y
90	Z
91	bracketleft
92	backslash
93	bracketright
94	asciicircum
95	underscore
96	quoteleft
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642915 994

Table E.2 (continued)

Index	Glyph ID
119	w
120	x
121	y
122	z
123	braceleft
124	bar
125	braceright
126	asciitilde
127	.notdef
128	.notdef
129	.notdef
130	.notdef
131	.notdef
132	notdef
133	notdef
134	.notdef
135	.notdef
136	.notdef
137	.notdef
138	.notdef
139	.notdef
140	.notdef
141	notdef
142	.notdef
143	.notdef
144	dotlessi
145	grave
146	acute
147	circumflex
148	tilde

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table E.2 (continued)

Index	Glyph ID
149	macron
150	breve
151	dotaccent
152	dieresis
153	.notdef
154	ring
155	cedilla
156	.notdef
157	hungarumlaut
158	ogonek
159	caron
160	space
161	exclamdown
162	cent
163	sterling
164	currency
165	yen
166	brokenbar
167	section
168	dieresis
169	copyright
170	ordfeminine
171	guillemotleft
172	logicalnot
173	hyphen
174	registered
175	macron
176	degree
177	plusminus
178	twosuperior

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642917 767

Table E.2 (continued)

Index	Glyph ID
179	threesuperior
180	acute
181	mu
182	paragraph
183	periodcentered
184	cedilla
185	onesuperior
186	ordmasculine
187	guillemotright
188	onequarter
189	onehalf
190	threequarters
191	questiondown
192	Agrave
193	Aacute
194	Acircumflex
195	Atilde
196	Adieresis
197	Aring
198	AE
199	Ccedilla
200	Egrave
201	Eacute
202	Ecircumflex
203	Edieresis
204	Igrave
205	Iacute
206	Icircumflex
207	Idieresis
208	Eth

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642918 6T3

Table E.2 (continued)

Index	Glyph ID
209	Ntilde
210	Ograve
211	Oacute
212	Ocircumflex
213	Otilde
214	Odieresis
215	multiply
216	Oslash
217	Ugrave
218	Uacute
219	Ucircumflex
220	Udieresis
221	Yacute
222	Thorn
223	germandbls
224	agrave
225	aacute
226	acircumflex
227	atilde
228	adieresis
229	aring
230	ae
231	ccedilla
232	egrave
233	eacute
234	ecircumflex
235	edieresis
236	igrave
237	iacute
238	icircumflex

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642919 53T

Table E.2 (concluded)

Index	Glyph ID
239	idieresis
240	eth
241	ntilde
242	ograve
243	oacute
244	ocircumflex
245	otilde
246	odieresis
247	divide
248	oslash
249	ugrave
250	uacute
251	ucircumflex
252	udieresis
253	yacute
254	thorn
255	ydieresis

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642920 251

Annex F
(informative)

ASN.1 Object Identifiers and Public Identifiers for Document Production Instructions defined by ISO/IEC 10175

F.1 ASN.1 Object Identifiers and Public Identifiers for Document Production Instructions defined by ISO/IEC 10175

This Informative Annex identifies Public Object Identifier values which are defined for Document Production Instructions by ISO/IEC 10175, the ISO/IEC 10175 attributes for which those values are defined, and the meaning of those attribute values.

For each Document Production Instruction Value defined, ISO/IEC 10175 specifies

— a public identifier as defined in ISO/IEC 9070;

— an object identifier value as defined in ISO/IEC 8824;

which identifies the value corresponding to the Object Name.

F.1.1 Document Production Instruction Values and Meanings

This clause identifies the Object Names corresponding to Public Object Identifier values which are defined by ISO/IEC 10175 for Document Production Instruction values and the meaning of those values.

F.1.1.1 Generic Public Identifier

ISO/IEC 10175 defines a generic Public Object Identifier to specify that no object is to be used or no process is required. It is used, for example, to specify that no break page is to be printed. The Object Names for which generic Public Object Identifier values are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F 1

Table F.1

Object Name	Meaning
dpi: generic: none	No object or process required

F.1.1.2 Output-bin

The Object Names for which Public Object Identifier values for the attribute *output-bin* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table F.2

Object Name	Meaning
dpi::output-bin::top	output bin best identified as the "top" bin
dpi::output-bin::middle	output bin best identified as the "middle" bin
dpi::output-bin::bottom	output bin best identified as the "bottom" bin
dpi::output-bin::side	output bin best identified as the "side" bin
dpi::output-bin::left	output bin best identified as the "left" bin
dpi::output-bin::right	output bin best identified as the "right" bin
dpi::output-bin::face-up	an output bin in which sheets are placed face up in the bin
dpi::output-bin::face-down	an output bin in which sheets are placed face down in the bin
dpi::output-bin::large-capacity	the output bin with the largest capacity
dpi::output-bin::private	the output bin which is identified with the print job originator

F.1.1.3 Medium

The Object Names for which Public Object Identifier values for the attribute *medium* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F 3.

Table F.3

Object Name	Meaning
dpi::medium::default	a system dependent default medium
dpi::medium::iso-a4::white	ISO Standard A4 white paper
dpi::medium::iso-a4::coloured	ISO Standard A4 paper
dpi::medium::iso-a4::transparent	ISO Standard A4 transparency material
dpi::medium::iso-a3::white	ISO Standard A3 white paper
dpi::medium::iso-a3::coloured	ISO Standard A3 colored paper
dpi::medium::iso-a5::white	ISO Standard A5 white paper
dpi::medium::iso-a5::coloured	ISO Standard A5 colored paper
dpi::medium::iso-b4::white	ISO Standard B4 white paper
dpi::medium::iso-b4::coloured	ISO Standard B4 colored paper
dpi::medium::iso-b5::white	ISO Standard B5 white paper
dpi::medium::iso-b5::coloured	ISO Standard B5 colored paper

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642922 024

Table F.3 (continued)

Object Name	Meaning
dpi::medium::na-letter::white	North American letter paper
dpi::medium::na-letter::coloured	North American letter paper
dpi::medium::na-letter::transparent	North American letter paper
dpi::medium::na-legal::white	North American legal paper
dpi::medium::na-legal::coloured	North American legal paper
dpi::medium::iso-b5-envelope	ISO Standard B5 paper
dpi::medium::iso-b4-envelope	ISO Standard B4 envelope
dpi::medium::iso-c3-envelope	ISO Standard C4 envelope
dpi::medium::iso-c4-envelope	ISO Standard C4 envelope
dpi::medium::iso-c5-envelope	ISO Standard C5 envelope
dpi::medium::iso-c6-envelope	ISO Standard C5 envelope
dpi::medium::iso-designated-long-envelope	ISO designated long envelope
dpi::medium::na-10x13-envelope	North American 10 in by 13 in envelope
dpi::medium::na-9x12	-whiteNorth American 9 in by 12 in paper
dpi::medium::na-number-10-envelope	North American number 10 envelope
dpi::medium::na-7x9-envelope	North American 7 in by 9 in envelope
dpi::medium::na-9x11-envelope	North American 9 in by 11 in envelope
dpi::medium::na-10x14-envelope	North American 10 in by 14 in envelope
dpi::medium::na-number-9-envelope	North American number 9 envelope
dpi::medium::na-6x9-envelope	North American 6 in by 9 in envelope
dpi::medium::na-10x15-envelope	North American 10 in by 15 in envelope
dpi::medium::jis-b4::white	**JIS** Standard B4 white paper
dpi::medium::jis-b4::coloured	**JIS** Standard B4 colored paper
dpi::medium::jis-b5::white	**JIS** Standard white B4 paper
dpi::medium::jis-b5::coloured	**JIS** Standard B4 colored paper
dpi::medium::iso-a0::white	ISO Standard white A0 paper
dpi::medium::iso-a1::white	ISO Standard white A1 paper
dpi::medium::iso-a2::white	ISO Standard A2 white paper
dpi::medium::iso-a6::white	ISO Standard A6 white paper
dpi::medium::iso-a7::white	ISO Standard A7 white paper

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table F.3 (concluded)

Object Name	Meaning
dpi::medium::iso-a8::white	ISO Standard A8 white paper
dpi::medium::iso-a9::white	ISO Standard A9 white paper
dpi::medium::iso-a10::white	ISO Standard **A10** white paper
dpi::medium::iso-b0::white	ISO Standard B0 white paper
dpi::medium::iso-b1::white	ISO Standard B1 white paper
dpi::medium::iso-b2::white	ISO Standard B2 white paper
dpi::medium::iso-b3::white	ISO Standard B3 white paper
dpi::medium::iso-b6::white	ISO Standard B6 white paper
dpi::medium::iso-b7::white	ISO Standard B7 white paper
dpi::medium::iso-b8::white	ISO Standard B8 white paper
dpi::medium::iso-b9::white	ISO Standard B9 white paper
dpi::medium::iso-b10::white	ISO Standard B10 white paper
dpi::medium::jis-b0::white	JIS Standard B0 white paper
dpi::medium::jis-b1::white	JIS Standard B1 white paper
dpi::medium::jis-b2::white	JIS Standard B2 white paper
dpi::medium::jis-b3::white	JIS Standard B3 white paper
dpi::medium::jis-b6::white	JIS Standard B6 white paper
dpi::medium::jis-b7::white	JIS Standard B7 white paper
dpi::medium::jis-b8::white	JIS Standard B8 white paper
dpi::medium::jis-b9::white	JIS Standard B9 white paper
dpi::medium::jis-b10::white	JIS Standard B10 white paper
dpi::medium::a	Engineering Medium, A-Size white paper
dpi::medium::b	Engineering Medium, B-Size white paper
dpi::medium::c	Engineering Medium, C-Size white paper
dpi::medium::d	Engineering Medium, D-Size white paper
dpi::medium::e	Engineering Medium, E-Size white paper

F.1.1.4 Medium-type

The Object Names for which Public Object Identifier values for the attribute *medium-type* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.4.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table F.4

Object Name	Meaning
dpi::medium::type::stationery	plain stationery
dpi::medium::type::transparency	transparency material
dpi::medium::type::envelope	envelopes
dpi::medium::type::envelope-plain	envelopes without windows
dpi::medium::type::envelope-window	envelopes with one or more transparent window(s)
dpi::medium::type::continuous-long	continuous form paper (e.g. computer printout paper) which is joined at the long edge
dpi::medium::type::continuous-short	continuous form paper (e.g. computer printout paper) which is joined at the short edge
dpi::medium::type::tab-stock	cut tab stock
dpi::medium::type::multi-part-form	multi-part forms
dpi::medium::type::labels	label stock
dpi::medium::type::multi-layer	multi-layer material

F.1.1.5 Medium-size

The Object Names for which Public Object Identifier values for the attribute *medium-size* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.5.

Table F.5

Object Name	Meaning
dpi::medium::size::iso-a0	ISO Standard A0 dimensions per ISO 216
dpi::medium::size::iso-a1	ISO Standard A1 dimensions per ISO 216
dpi::medium::size::iso-a2	ISO Standard A2 dimensions per ISO 216
dpi::medium::size::iso-a3	ISO Standard A3 dimensions per ISO 216
dpi::medium::size::iso-a4	ISO Standard A4 dimensions per ISO 216
dpi::medium::size::iso-a5	ISO Standard A5 dimensions per ISO 216
dpi::medium::size::iso-a6	ISO Standard A6 dimensions per ISO 216
dpi::medium::size::iso-a7	ISO Standard A7 dimensions per ISO 216
dpi::medium::size::iso-a8	ISO Standard A8 dimensions per ISO 216
dpi::medium::size::iso a9	ISO Standard A9 dimensions per ISO 216
dpi::medium::size::iso-a10	ISO Standard A10 dimensions per ISO 216

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642925 833

Table F.5 (continued)

Object Name	Meaning
dpi::medium::size::iso-b0	ISO Standard B0 dimensions per ISO 216
dpi::medium::size::iso-b1	ISO Standard B1 dimensions per ISO 216
dpi::medium::size::iso-b2	ISO Standard B2 dimensions per ISO 216
dpi::medium::size::iso-b3	ISO Standard B3 dimensions per ISO 216
dpi::medium::size::iso-b4	ISO Standard B4 dimensions per ISO 216
dpi::medium::size::iso-b5	ISO Standard B5 dimensions per ISO 216
dpi::medium::size::iso-b6	ISO Standard B6 dimensions per ISO 216
dpi::medium::size::iso-b7	ISO Standard B7 dimensions per ISO 216
dpi::medium::size::iso-b8	ISO Standard B8 dimensions per ISO 216
dpi::medium::size::iso-b9	ISO Standard B9 dimensions per ISO 216
dpi::medium::size::iso-b10	ISO Standard B10 dimensions per ISO 216
dpi::medium::size::iso-c3	ISO Standard C3 dimensions per ISO 269
dpi::medium::size::iso-c4	ISO Standard C4 dimensions per ISO 269
dpi::medium::size::iso-c5	ISO Standard C5 dimensions per ISO 269
dpi::medium::size::iso-c6	ISO Standard C6 dimensions per ISO 269
dpi::medium::size::iso-designated-long	ISO Standard designated long per ISO 269
dpi::medium::size::jis-b0	JIS Standard B0 dimensions
dpi::medium::size::jis-b1	JIS Standard B1 dimensions
dpi::medium::size::jis-b2	JIS Standard B2 dimensions
dpi::medium::size::jis-b3	JIS Standard B3 dimensions
dpi::medium::size::jis-b4	JIS Standard B4 dimensions
dpi::medium::size::jis-b5	JIS Standard B5 dimensions
dpi::medium::size::jis-b6	JIS Standard B6 dimensions
dpi::medium::size::jis-b7	JIS Standard B7 dimensions
dpi::medium::size::jis-b8	JIS Standard B8 dimensions
dpi::medium::size::jis-b9	JIS Standard B9 dimensions
dpi::medium::size::jis-b10	JIS Standard B10 dimensions
dpi::medium::size::na-letter	North American letter (216, 280)
dpi::medium::size::na-legal	North American legal (216, 356)
dpi::medium::size::10x13-envelope	North American 10 by 13 (254, 330)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642926 77T

Table F.5 (concluded)

Object Name	Meaning
dpi::medium::size::9x12-envelope	North American 9 in by 12 in (229, 305)
dpi::medium::size::na-number-10-envelope	North American envelope
dpi::medium::size::na-7x9-envelope	North American envelope
dpi::medium::size::na-9x11envelope	North American envelope
dpi::medium::size::na-10x14envelope	North American envelope
dpi::medium::size::na-number-9-envelope	North American envelope
dpi::medium::size::na-6x9-envelope	North American envelope
dpi::medium::size::na-10x15-envelope	North American envelope
dpi::medium::size::a	Engineering medium, A-size
dpi::medium::size::b	Engineering medium, B-size
dpi::medium::size::c	Engineering medium, C-size
dpi::medium::size::d	Engineering medium, D-size
dpi::medium::size::e	Engineering medium, E-size

F.1.1.6 Medium-colour

The Object Names for which Public Object Identifier values for the attribute *medium-colour* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F 6.

Table F.6

Object Name	Meaning
dpi::medium::colour::white	white
dpi::medium::colour::pink	pink
dpi::medium::colour::yellow	light yellow
dpi::medium::colour::buff	buff
dpi::medium::colour::goldenrod	goldenrod
dpi::medium::colour::blue	light blue
dpi::medium::colour::green	light green
dpi::medium::colour::transparent-untinted	transparent untinted medium

F.1.1.7 Medium-tooth

The Object Names for which Public Object Identifier values for the attribute *medium-tooth* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F 7

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642927 606

Table F.7

Object Name	Meaning
dpi::medium::tooth::fine	fine tooth paper
dpi::medium::tooth::coarse	coarse tooth paper

F.1.1.8 Medium-grain

The Object Names for which Public Object Identifier values for the attribute *medium-grain* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.8

Table F.8

Object Name	Meaning
dpi::medium::grain::long	the grain shall be in the direction of the long edge of the medium
dpi::medium::grain::short	the grain shall be in the direction of the short edge of the medium

F.1.1.9 Auxiliary-sheet

The Object Names for which Public Object Identifier values for the attribute *auxiliary-sheet* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.9.

Table F.9

Object Name	Meaning
dpi::auxiliary-sheet::start-sheet::brief	A brief form start sheet which contains only minimal information
dpi::auxiliary-sheet::start-sheet::full	A verbose form start sheet containing maximal information
dpi::auxiliary-sheet::separator-sheet::brief	A brief form separator sheet which contains only minimal information
dpi::auxiliary-sheet::separator-sheet::full	A verbose form separator sheet containing maximal information
dpi::auxiliary-sheet::end-sheet::brief	A brief form end sheet which contains only minimal information
dpi::auxiliary-sheet::end-sheet::full	A verbose form end sheet containing maximal information
dpi::auxiliary-sheet::blank	A blank auxiliary sheet
dpi::auxiliary-sheet::slipsheet	A blank auxiliary sheet, normally of a distinctive color

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642928 542

F.1.1.10 Stitching

The Object Names for which Public Object Identifier values for the attribute *stitching* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.10.

Table F.10

Object Name	Meaning
dpi..stitching::default	a system dependent stitching Finishing Process
dpi· stitching :staple-top-left	a single staple is placed in the top left corner of the finished document relative to the Reference Coordinate System
dpi::stitching··staple-bottom-left	a single staple is placed in the upper left corner of the finished document relative to the Reference Coordinate System
dpi::stitching. staple-top-right	a single staple is placed in the top right corner of the finished document relative to the Reference Coordinate System
dpi:.stitching··staple-bottom-right	a single staple is placed in the bottom right corner of the finished document relative to the Reference Coordinate System
dpi· stitching..saddle-stitch	a continuous line of stitching is between the bottom and top edges in the middle of the document
dpi··stitching ·edge-stitch	a continuous line of stitching is placed near the left edge of the document
dpi .stitching .edge-stitch-right	a continuous line of stitching is placed near the right edge of the document
dpi :stitching: edge-stitch-top	a continuous line of stitching is placed near the top edge of the document

F.1.1.11 Stitch-type

The Object Names for which Public Object Identifier values for the attribute *stitch-type* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.11.

Table F.11

Object Name	Meaning
dpi::stitching::type··default	a system dependent type of stitching
dpi::stitching··type.:staple	one or more staples applied at the specified head positions
dpi.:stitching ·type· stitch	one or more stitches applied at the specified head positions
dpi..stitching··type. continuous	a continuous line of stitching along the process axis

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642929 489

F.1.1.12 Binding

The Object Names for which Public Object Identifier values for the attribute *binding-type* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.12.

Table F.12

Object Name	Meaning
dpi::binding::default	a system dependent binding Finishing Process

F.1.1.13 Binding-type

The Object Names for which Public Object Identifier values for the attribute *binding-type* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.13

Table F.13

Object Name	Meaning
dpi::binding::type::default	a system dependent type of binding is applied
dpi::binding::type::tape	a tape binding is applied
dpi::binding::type::plastic	a plastic binding is applied
dpi::binding::type::velo	a velo binding is applied
dpi::bindingtype::perfect	a binding commonly referred to as a "perfect" binding is applied
dpi::binding::type::spiral	a spiral binding is applied

F.1.1.14 Binding-colour

The Object Names for which Public Object Identifier values for the attribute *binding-colour* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.14

Table F.14

Object Name	Meaning
dpi::binding::colour::default	a system dependent color of binding material
dpi::binding::colour::black	black binding material
dpi::binding::colour::blue	blue binding material
dpi::binding::colour::gray	gray binding material
dpi::binding::colour::brown	brown binding material

F.1.1.15 Perforation

The Object Names for which Public Object Identifier values for the attribute *perforation* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.15.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table F.15

Object Name	Meaning
dpi::perforation..default	a system dependent perforation Finishing Process

F.1.1.16 Perforation-type

The Object Names for which Public Object Identifier values for the attribute *perforation-type* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.16

Table F.16

Object Name	Meaning
dpi perforation type::default	a system dependent type of perforation

F.1.1.17 Trimming

The Object Names for which Public Object Identifier values for the attribute *trimming* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.17.

Table F.17

Object Name	Meaning
dpi trimming default	a system dependent trimming Finishing Process

F.1.1.18 Punching

The Object Names for which Public Object Identifier values for the attribute *punching* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F 18.

Table F.18

Object Name	Meaning
dpi punching .default	a system dependent punching Finishing Process

F.1.1.19 Slitting

The Object Names for which Public Object Identifier values for the attribute *slitting* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.19.

Table F.19

Object Name	Meaning
dpi::slitting::default	a system dependent slitting Finishing Process

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642931 037

F.1.1.20 Covers

The Object Names for which Public Object Identifier values for the attribute *covers* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.20

Table F.20

Object Name	Meaning
dpi::covers::default	system dependent covers are applied

F.1.1.21 Cover-name

The Object Names for which Public Object Identifier values for the attribute *cover-name* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.21.

Table F.21

Object Name	Meaning
dpi::cover-name::default	system dependent cover is used

F.1.1.22 Folding

The Object Names for which Public Object Identifier values for the attribute *folding* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.22

Table F.22

Object Name	Meaning
dpi::folding::default	a system dependent folding Finishing Process

F.1.1.23 Plex

The Object Names for which Public Object Identifier values for the attribute *plex* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.23.

Table F.23

Object Name	Meaning
dpi::plex::simplex	Specifies image placement suitable for the placement of a single page image on the obverse side (front side) of each instance of the medium
dpi::plex::duplex	Specifies image placement suitable for the placement of two page images on each instance of the medium. When sides = 2, the Reference Coordinate Systems on the obverse and reverse (back) sides are positioned on the medium such that the bottom edge (*x* -axis of the Reference Coordinate System) of both surfaces is the same edge of the medium

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642932 T73

Table F.23 (concluded)

Object Name	Meaning
dpi::plex.:tumble	Specifies the placement of two page images on each instance of the medium. When sides = 2, the Reference Coordinate Systems on the obverse and reverse (back) sides are positioned on the medium such that the bottom edge (*x* -axis of the Reference Coordinate System) of the obverse surface is the opposite edge of the medium from that of the reverse surface

F.1.1.24 Output

The Object Names for which Public Object Identifier values for the attribute *output* are defined by ISO/IEC 10175, and the meaning of those values, are as shown in table F.24.

Table F.24

Object Name	Meaning
dpi::output:.default-stacking	a system dependent output stacking

F.1.2 ASN.1 Object Identifiers and Public Identifiers

Each Document Production Instruction Value defined by ISO/IEC 10175 is assigned a public identifier as defined in ISO 9070. The canonical character string form of the public identifier which corresponds to the Object Name *object-name* consists of the registered owner prefix

ISO/IEC 10175-1

followed by "//" and the character sequence *object-name*.

Each public identifier has a semantically equivalent **SGML** Formal Public Identifier. **SGML** Formal Public Identifiers are defined in ISO/IEC 8879.

F.1.3 Object Identifier Value References

Each Document Production Instruction Value defined by ISO/IEC 10175 is assigned an ASN.1 object identifier value. The ASN 1 value reference which designates the object identifier value which corresponds to each Object Name is listed below.

F.1.3.1 Generic Object Identifier Value References

Table F.25

Object Name	Object Identifier Value Reference
dpi: generic:·none	id-val-generic-none

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642933 90T

F.1.3.2 Output-bin Public Object Identifier Value References

Table F.26

Object Name	Object Identifier Value Reference
dpi::output-bin::top	*id-val-output-bin-top*
dpi::output-bin::middle	id-val-output-bin-middle
dpi::output-bin::bottom	id-val-output-bin-bottom
dpi::output-bin::side	id-val-output-bin-side
dpi::output-bin::face-up	id-val-output-bin-face-up
dpi::output-bin::left	id-val-output-bin-left
dpi::output-bin::right	id-val-output-bin-right
dpi::output-bin::large-capacity	id-val-output-bin-large-capacity
dpi::output-bin::private	id-val-output-bin-private
dpi::output-bin::face-down	id-val-output-bin-face-down

F.1.3.3 Medium Object Identifier Value References

Table F.27

Object Name	Object Identifier Value Reference
dpi::medium::default	id-val-medium-default
dpi::medium::iso-a4::white	id-val-medium-iso-a4-white
dpi::medium::iso-a4::coloured	id-val-medium-iso-a4-coloured
dpi::medium::iso-a4::transparent	id-val-medium-iso-a4-transparent
dpi::medium::iso-a3::white	id-val-medium-iso-a3-white
dpi::medium::iso-a3::coloured	id-val-medium-iso-a3-coloured
dpi::medium::iso-a5::white	id-val-medium-iso-a5-white
dpi::medium::iso-a5::coloured	id-val-medium-iso-a5-coloured
dpi::medium::iso-b4::white	id-val-medium-iso-b4-white
dpi::medium::iso-b4::coloured	id-val-medium-iso-b4-coloured
dpi::medium::iso-b5::white	id-val-medium-iso-b5-white
dpi::medium::iso-b5::coloured	id-val-medium-iso-b5-coloured
dpi::medium::na-letter::white	id-val-medium-na-letter-white
dpi::medium::na-letter::coloured	id-val-medium-na-letter-coloured

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642934 846

Table F.27 (continued)

Object Name	Object Identifier Value Reference
dpi::medium::na-letter::transparent	id-val-medium-na-letter-transparent
dpi::medium::na-legal::white	id-val-medium-na-legal-white
dpi::medium::na-legal::coloured	id-val-medium-na-legal-coloured
dpi::medium::iso-b5-envelope	id-val-medium-iso-b5-envelope
dpi::medium::iso-b4-envelope	id-val-medium-iso-b4-envelope
dpi::medium::iso-c3-envelope	id-val-medium-iso-c3-envelope
dpi::medium::iso-c4-envelope	id-val-medium-iso-c4-envelope
dpi::medium::iso-c5-envelope	id-val-medium-iso-c5-envelope
dpi::medium::iso-c6-envelope	id-val-medium-iso-c6-envelope
dpi::medium::iso-designated-long-envelope	id-val-medium-iso-designated-long-envelope
dpi::medium::na-10x13-envelope	id-val-medium-na-10x13-envelope
dpi::medium::na-9x12 -envelope	id-val-medium-na-9x12-envelope
dpi::medium::na-number-10-envelope	id-val-medium-na-number-10-envelope
dpi::medium::na-7x9-envelope	id-val-medium-na-7x9-envelope
dpi::medium::na-9x11-envelope	id-val-medium-na-9x11-envelope
dpi::medium::na-10x14 envelope	id-val-medium-na-10x14-envelope
dpi::medium::na-number-9-envelope	id-val-medium-na-number-9-envelope
dpi::medium::na-6x9-envelope	id-val-medium-na-6x9-envelope
dpi::medium::na-10x15-envelope	id-val-medium-na-10x15-envelope
dpi::medium::jis-b4::white	id-val-medium-jis-b4-white
dpi::medium::jis-b4::coloured	id-val-medium-jis-b4-coloured
dpi::medium::jis-b5::white	id-val-medium-jis-b5-white
dpi::medium::jis-b5::coloured	id-val-medium-jis-b5-coloured
dpi::medium::iso-a0::white	id-val-medium-iso-a0-white
dpi::medium::iso-a1::white	id-val-medium-iso-a1-white
dpi::medium::iso-a2::white	id-val-medium-iso-a2-white
dpi::medium::iso-a6::white	id-val-medium-iso-a6-white
dpi::medium::iso-a7::white	id-val-medium-iso-a7-white
dpi::medium::iso-a8::white	id-val-medium-iso-a8-white
dpi::medium::iso-a9::white	id-val-medium-iso-a9-white

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642935 782

Table F.27 (concluded)

Object Name	Object Identifier Value Reference
dpi::medium::iso-a10::white	id-val-medium-iso-a10-white
dpi::medium::iso-b0::white	id-val-medium-iso-b0-white
dpi::medium::iso-b1::white	id-val-medium-iso-b1-white
dpi::medium::iso-b2::white	id-val-medium-iso-b2-white
dpi::medium::iso-b3::white	id-val-medium-iso-b3-white
dpi::medium::iso-b6::white	id-val-medium-iso-b6-white
dpi::medium::iso-b7::white	id-val-medium-iso-b7-white
dpi::medium::iso-b8::white	id-val-medium-iso-b8-white
dpi::medium::iso-b9::white	id-val-medium-iso-b9-white
dpi::medium::iso-b10::white	id-val-medium-iso-b10-white
dpi::medium::jis-b0::white	id-val-medium-jis-b0-white
dpi::medium::jis-b1::white	id-val-medium-jis-b1-white
dpi::medium::jis-b2::white	id-val-medium-jis-b2-white
dpi::medium::jis-b3::white	id-val-medium-jis-b3-white
dpi::medium::jis-b6::white	id-val-medium-jis-b6-white
dpi::medium::jis-b7::white	id-val-medium-jis-b7-white
dpi::medium::jis-b8::white	id-val-medium-jis-b8-white
dpi::medium::jis-b9::white	id-val-medium-jis-b9-white
dpi::medium::jis-b10::white	id-val-medium-jis-b10-white
dpi::medium::a	id-val-medium-a
dpi::medium::b	id-val-medium-b
dpi::medium::c	id-val-medium-c
dpi::medium::d	id-val-medium-d
dpi::medium::e	id-val-medium-e

F.1.3.4 Medium-type Object Identifer Value References

Table F.28

Object Name	Object Identifier Value Reference
dpi::medium::type::stationery	id-val-medium-type-stationery
dpi::medium::type::transparency	id-val-medium-type-transparency

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642936 619

Table F.28 (concluded)

Object Name	Object Identifier Value Reference
dpi::medium::type::envelope	id-val-medium-type-envelope
dpi::medium::type::envelope-plain	id-val-medium-type-envelope-plain
dpi::medium::type::envelope-window	id-val-medium-type-envelope-window
dpi::medium::type::continuous-long	id-val-medium-type-continuous-long
dpi::medium::type::continuous-short	id-val-medium-type-continuous-short
dpi::medium::type::tab-stock	id-val-medium-type-tab-stock
dpi::medium::type::multi-part-form	id-val-medium-type-multi-part-form
dpi::medium::type::labels	id-val-medium-type-labels
dpi::medium::type::multi-layer	id-val-medium-type-multi-layer

F.1.3.5 Medium-size Object Identifer Value References

Table F.29

Object Name	Object Identifier Value Reference
dpi::medium::size::iso-a0	id-val-medium-size-iso-a0
dpi::medium::size::iso-a1	id-val-medium-size-iso-a1
dpi::medium::size::iso-a2	id-val-medium-size-iso-a2
dpi::medium::size::iso-a3	id-val-medium-size-iso-a3
dpi::medium::size::iso-a4	id-val-medium-size-iso-a4
dpi::medium::size::iso-a5	id-val-medium-size-iso-a5
dpi::medium::size::iso-a6	id-val-medium-size-iso-a6
dpi::medium::size::iso-a7	id-val-medium-size-iso-a7
dpi::medium::size::iso-a8	id-val-medium-size-iso-a8
dpi::medium::size::iso-a9	id-val-medium-size-iso-a9
dpi::medium::size::iso-a10	id-val-medium-size-iso-a10
dpi::medium::size::iso-b0	id-val-medium-size-iso-b0
dpi::medium::size::iso-b1	id-val-medium-size-iso-b1
dpi::medium::size::iso-b2	id-val-medium-size-iso-b2
dpi::medium::size::iso-b3	id-val-medium-size-iso-b3
dpi::medium::size::iso-b4	id-val-medium-size-iso-b4
dpi::medium::size::iso-b5	id-val-medium-size-iso-b5

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642937 555

Table F.29 (continued)

Object Name	Object Identifier Value Reference
dpi::medium::size::iso-b6	id-val-medium-size-iso-b6
dpi::medium::size::iso-b7	id-val-medium-size-iso-b7
dpi::medium::size::iso-b8	id-val-medium-size-iso-b8
dpi::medium::size::iso-b9	id-val-medium-size-iso-b9
dpi::medium::size::iso-b10	id-val-medium-size-iso-b10
dpi::medium::size::iso-c3	id-val-medium-size-iso-c3
dpi::medium::size::iso-c4	id-val-medium-size-iso-c4
dpi::medium::size::iso-c5	id-val-medium-size-iso-c5
dpi::medium::size::iso-c6	id-val-medium-size-iso-c6
dpi::medium::size::iso-designated-long	id-val-medium-size-iso-designated-long
dpi::medium::size::jis-b0	id-val-medium-size-jis-b0
dpi::medium::size::jis-b1	id-val-medium-size-jis-b1
dpi::medium::size::jis-b2	id-val-medium-size-jis-b2
dpi::medium::size::jis-b3	id-val-medium-size-jis-b3
dpi::medium::size::jis-b4	id-val-medium-size-jis-b4
dpi::medium::size::jis-b5	id-val-medium-size-jis-b5
dpi::medium::size::jis-b6	id-val-medium-size-jis-b6
dpi::medium::size::jis-b7	id-val-medium-size-jis-b7
dpi::medium::size::jis-b8	id-val-medium-size-jis-b8
dpi::medium::size::jis-b9	id-val-medium-size-jis-b9
dpi::medium::size::jis-b10	id-val-medium-size-jis-b10
dpi::medium::size::na-letter	id-val-medium-size-na-letter
dpi::medium::size::na-legal	id-val-medium-size-na-legal
dpi::medium::size::10x13-envelope	id-val-medium-size-na-10x13-envelope
dpi::medium::size::9x12-envelope	id-val-medium-size-na-9x12-envelope
dpi::medium::size::na-number-10-envelope	id-val-medium-size-na-number-10-envelope
dpi::medium::size::na-7x9-envelope	id-val-medium-size-na-7x9-envelope
dpi::medium::size::na-9x11-envelope	id-val-medium-size-na-9x11-envelope
dpi::medium::size::na-10x14-envelope	id-val-medium-size-na-10x14-envelope
dpi::medium::size::na-number-9-envelope	id-val-medium-size-na-number-9-envelope

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642938 491

Table F.29 (concluded)

Object Name	Object Identifier Value Reference
dpi::medium::size::na-6x9-envelope	id-val-medium-size-na-6x9-envelope
dpi::medium::size::na-10x15-envelope	id-val-medium-size-na-10x15-envelope
dpi::medium::size::a	id-val-medium-size-a
dpi::medium::size::b	id-val-medium-size-b
dpi::medium::size::c	id-val-medium-size-c
dpi::medium::size::d	id-val-medium-size-d
dpi::medium::size::e	id-val-medium-size-e

F.1.3.6 Medium-colour Object Identifer Value References

Table F.30

Object Name	Object Identifier Value Reference
dpi::medium::colour::white	id-val-medium-colour-white
dpi::medium::colour::pink	id-val-medium-colour-pink
dpi::medium::colour::yellow	id-val-medium-colour-yellow
dpi::medium::colour::buff	id-val-medium-colour-buff
dpi::medium::colour::goldenrod	id-val-medium-colour-goldenrod
dpi::medium::colour::blue	id-val-medium-colour-blue
dpi::medium::colour::green	id-val-medium-colour-green
dpi::medium::colour::transparent-untinted	id-val-medium-colour-transparent-untinted

F.1.3.7 Medium-tooth Object Identifer Value References

Table F.31

Object Name	Object Identifier Value Reference
dpi::medium::tooth::fine	id-val-medium-tooth-fine
dpi::medium::tooth::coarse	id-val-medium-tooth-coarse

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642939 328

F.1.3.8 Medium-grain Object Identifer Value References

Table F.32

Object Name	Object Identifier Value Reference
dpi::medium::grain::long	id-val-medium-grain-long
dpi::medium::grain::short	id-val-medium-grain-short

F.1.3.9 Auxiliary-sheet Object Identifer Value References

Table F.33

Object Name	Object Identifier Value Reference
dpi::auxiliary-sheet::start-sheet::brief	id-val-auxiliary-sheet-start-sheet-brief
dpi::auxiliary-sheet::start-sheet::full	id-val-auxiliary-sheet-start-sheet-full
dpi::auxiliary-sheet::separator-sheet::brief	id-val-auxiliary-sheet-separator-sheet-brief
dpi::auxiliary-sheet::separator-sheet::full	id-val-auxiliary-sheet-separator-sheet-full
dpi::auxiliary-sheet::end-sheet::brief	id-val-auxiliary-sheet-end-sheet-brief
dpi::auxiliary-sheet::end-sheet::full	id-val-auxiliary-sheet-end-sheet-full
dpi::auxiliary-sheet::blank	id-val-auxiliary-sheet-blank
dpi::auxiliary-sheet::slipsheet	id-val-auxiliary-sheet-slipsheet

F.1.3.10 Stitching Object Identifer Value References

Table F.34

Object Name	Object Identifier Value Reference
dpi::stitching::default	id-val-stitching-default
dpi::stitching::staple	id-val-stitching-staple
dpi::stitching::staple-top-left	id-val-stitching-staple-top-left
dpi::stitching::staple-bottom-left	id-val-stitching-staple-bottom-left
dpi::stitching::staple-top-right	id-val-stitching-staple-top-right
dpi::stitching::staple-bottom-right	id-val-stitching-staple-bottom-right
dpi::stitching::saddle-stitch	id-val-stitching-saddle-stitch
dpi::stitching::edge-stitch	id-val-stitching-edge-stitch
dpi::stitching::edge-stitch-right	id-val-stitching-edge-stitch-right
dpi::stitching::edge-stitch-top	id-val-stitching-edge-stitch-top

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642940 04T

F.1.3.11 Stitch-type Object Identifer Value References

Table F.35

Object Name	Object Identifier Value Reference
dpi::stitching::type::default	id-val-stitching-type-default
dpi::stitching::type::staple	id-val-stitching-type-staple
dpi::stitching::type::stitch	id-val-stitching-type-stitch
dpi::stitching::type::continuous	id-val-stitching-type-continuous

F.1.3.12 Binding Object Identifer Value References

Table F.36

Object Name	Object Identifier Value Reference
dpi::binding::default	id-val-binding-default

F.1.3.13 Binding-type Object Identifer Value References

Table F.37

Object Name	Object Identifier Value Reference
dpi::binding::type::default	id-val-binding-type-default
dpi::binding::type::tape	id-val-binding-type-tape
dpi::binding::type ::plastic	id-val-binding-type-plastic
dpi::binding ::type::velo	id-val-binding-type-velo
dpi::bindingtype::perfect	id-val-binding-type-perfect
dpi::binding::type::spiral	id-val-binding-type-spiral

F.1.3.14 Binding-colour Object Identifer Value References

Table F.38

Object Name	Object Identifier Value Reference
dpi::binding::colour::default	id-val-binding-colour-default
dpi::binding::colour::black	id-val-binding-colour-black
dpi::binding::colour::blue	id-val-binding-colour-blue

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table F.38 (concluded)

Object Name	Object Identifier Value Reference
dpi::binding::colour::gray	id-val-binding-colour-gray
dpi::binding::colour::brown	id-val-binding-colour-brown

F.1.3.15 Perforation Object Identifer Value References

Table F.39

Object Name	Object Identifier Value Reference
dpi::perforation::default	id-val-*perforation* -default

F.1.3.16 Perforation-type Object Identifer Value References

Table F.40

Object Name	Object Identifier Value Reference
dpi::perforation-type::default	id-val-perforation-type-default

F.1.3.17 Trimming Object Identifer Value References

Table F.41

Object Name	Object Identifier Value Reference
dpi::trimming::default	id-val-trimming-default

F.1.3.18 Punching Object Identifer Value References

Table F.42

Object Name	Object Identifier Value Reference
dpi::punching::default	id-val-punching-default

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4851903 0642942 912

F.1.3.19 Slitting Object Identifer Value References

Table F.43

Object Name	Object Identifier Value Reference
dpi::slitting::default	id-val-slitting-default

F.1.3.20 Covers Object Identifer Value References

Table F.44

Object Name	Object Identifier Value Reference
dpi::covers::default	id-val-covers-default

F.1.3.21 Cover-name Object Identifer Value References

Table F.45

Object Name	Object Identifier Value Reference
dpi::cover-name::default	id-val-cover-name-default

F.1.3.22 Folding Object Identifer Value References

Table F.46

Object Name	Object Identifier Value Reference
dpi::folding::default	id-val-folding-default

F.1.3.23 Plex Object Identifer Value References

Table F.47

Object Name	Object Identifier Value Reference
dpi::plex::simplex	id-val-plex-simplex
dpi::plex::duplex	id-val-plex-duplex
dpi::plex::tumble	id-val-plex-tumble

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

F.1.3.24 Stacking Object Identifer Value References

Table F.48

Object Name	Object Identifier Value Reference
dpi::output::stacking::default	id-val-output-stacking-default

F.1.4 ASN.1 Object Identifier Definitions

The following ASN.1 Module contains the object identifier value definitions for Document Production Instruction Values from ISO/IEC 10175-1:1993. The definitions in in this module are informative only· the actual definition of the object identifier values is the normative text of ISO/IEC 10175.

```
DEFINITIONS                                    ::= BEGIN
ID                                             ::= OBJECT IDENTIFIER
-- Basic Object Identifier Definitions
id-dpa                          ID  ::= { iso(1) standard(0) 10175 part-1(1) }
id-vc                           ID  ::= { id-dpa 6 }
                                              --attribute value classes--

-- Value Classes
id-vc-generic                                  ID  ::= { id-vc  2 }
id-vc-output-bin                               ID  ::= { id-vc  33 }
id-vc-input-tray                               ID  ::= { id-vc  34 }
id-vc-medium                                   ID  ::= { id-vc  50 }
id-vc-medium-type                              ID  ::= { id-vc  52 }
id-vc-medium-size                              ID  ::= { id-vc  53 }
id-vc-medium-colour                            ID  ::= { id-vc  54 }
id-vc-medium-tooth                             ID  ::= { id-vc  5 }
id-vc-medium-grain                             ID  ::= { id-vc  6 }
id-vc-auxiliary-sheet                          ID  ::= { id-vc  85 }
id-vc-finishing                                ID  ::= { id-vc  90 }
id-vc-stitching                                ID  ::= { id-vc  91 }
id-vc-stitching-type                           ID  ::= { id-vc  93 }
id-vc-binding                                  ID  ::= { id-vc  92 }
id-vc-binding-type                             ID  ::= { id-vc  94 }
id-vc-binding-colour                           ID  ::= { id-vc  95 }
id-vc-perforation                              ID  ::= { id-vc  97 }
id-vc-perforation-type                         ID  ::= { id-vc  96 }
id-vc-trimming                                 ID  ::= { id-vc  98 }
id-vc-punching                                 ID  ::= { id-vc  99 }
id-vc-slitting                                 ID  ::= { id-vc  110 }
id-vc-covers                                   ID  ::= { id-vc  111 }
id-vc-cover-name                               ID  ::= { id-vc  112 }
id-vc-folding                                  ID  ::= { id-vc  113 }
id-vc-plex                                     ID  ::= { id-vc  101 }
id-vc-output                                   ID  ::= { id-vc  115 }
-- Generic Object Identifier Values
id-val-generic-none                            ID  ::= { id-vc-generic  0 }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642944 795


```
-- Output Bin Object Identifier Values
id-val-output-bin-top                          ID  ::= { id-vc-output-bin  0 }
id-val-output-bin-middle                       ID  ::= { id-vc-output-bin  1 }
id-val-output-bin-bottom                       ID  ::= { id-vc-output-bin  2 }
id-val-output-bin-side                         ID  ::= { id-vc-output-bin  3 }
id-val-output-bin-face-up                      ID  ::= { id-vc-output-bin  4 }
id-val-output-bin-left                         ID  ::= { id-vc-output-bin  5 }
id-val-output-bin-right                        ID  ::= { id-vc-output-bin  6 }
id-val-output-bin-large-capacity               ID  ::= { id-vc-output-bin  7 }
id-val-output-bin-private                      ID  ::= { id-vc-output-bin  8 }
id-val-output-bin-face-down                    ID  ::= { id-vc-output-bin  9 }
-- Medium Object Identifier Values
id-val-medium-default                          ID  ::= { id-vc-medium  1000 }
id-val-medium-iso-a4-white                     ID  ::= { id-vc-medium  0 }
id-val-medium-iso-a4-coloured                  ID  ::= { id-vc-medium  1 }
id-val-medium-iso-a4-transparent               ID  ::= { id-vc-medium  2 }
id-val-medium-iso-a3-white                     ID  ::= { id-vc-medium  10 }
id-val-medium-iso-a3-coloured                  ID  ::= { id-vc-medium  11 }
id-val-medium-iso-a5-white                     ID  ::= { id-vc-medium  20 }
id-val-medium-iso-a5-coloured                  ID  ::= { id-vc-medium  21 }
id-val-medium-iso-b4-white                     ID  ::= { id-vc-medium  30 }
id-val-medium-iso-b4-coloured                  ID  ::= { id-vc-medium  31 }
id-val-medium-iso-b5-white                     ID  ::= { id-vc-medium  40 }
id-val-medium-iso-b5-coloured                  ID  ::= { id-vc-medium  41 }
id-val-medium-na-letter-white                  ID  ::= { id-vc-medium  50 }
id-val-medium-na-letter-coloured               ID  ::= { id-vc-medium  51 }
id-val-medium-na-letter-transparent            ID  ::= { id-vc-medium  52 }
id-val-medium-na-legal-white                   ID  ::= { id-vc-medium  60 }
id-val-medium-na-legal-coloured                ID  ::= { id-vc-medium  61 }
id-val-medium-iso-b5-envelope                  ID  ::= { id-vc-medium  73 }
id-val-medium-iso-b4-envelope                  ID  ::= { id-vc-medium  83 }
id-val-medium-iso-c3-envelope                  ID  ::= { id-vc-medium  88 }
id-val-medium-iso-c4-envelope                  ID  ::= { id-vc-medium  93 }
id-val-medium-iso-c5-envelope                  ID  ::= { id-vc-medium  103 }
id-val-medium-iso-c6-envelope                  ID  ::= { id-vc-medium  108 }
id-val-medium-iso-designated-long-envelope     ID  ::= { id-vc-medium  113 }
id-val-medium-na-10x13-envelope                ID  ::= { id-vc-medium  123 }
id-val-medium-na-9x12-envelope                 ID  ::= { id-vc-medium  133 }
id-val-medium-na-number-10-envelope            ID  ::= { id-vc-medium  143 }
id-val-medium-na-7x9-envelope                  ID  ::= { id-vc-medium  144 }
id-val-medium-na-9x11-envelope                 ID  ::= { id-vc-medium  145 }
id-val-medium-na-10x14-envelope                ID  ::= { id-vc-medium  146 }
id-val-medium-na-number-9-envelope             ID  ::= { id-vc-medium  147 }
id-val-medium-na-6x9-envelope                  ID  ::= { id-vc-medium  148 }
id-val-medium-na-10x15-envelope                ID  ::= { id-vc-medium  149 }
id-val-medium-jis-b4-white                     ID  ::= { id-vc-medium  150 }
id-val-medium-jis-b4-coloured                  ID  ::= { id-vc-medium  151 }
id-val-medium-jis-b5-white                     ID  ::= { id-vc-medium  160 }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
id-val-medium-jis-b5-coloured                ID  ::= { id-vc-medium  161 }
id-val-medium-iso-a0-white                   ID  ::= { id-vc-medium  200 }
id-val-medium-iso-a1-white                   ID  ::= { id-vc-medium  201 }
id-val-medium-iso-a2-white                   ID  ::= { id-vc-medium  202 }
id-val-medium-iso-a6-white                   ID  ::= { id-vc-medium  206 }
id-val-medium-iso-a7-white                   ID  ::= { id-vc-medium  207 }
id-val-medium-iso-a8-white                   ID  ::= { id-vc-medium  208 }
id-val-medium-iso-a9-white                   ID  ::= { id-vc-medium  209 }
id-val-medium-iso-a10-white                  ID  ::= { id-vc-medium  210 }
id-val-medium-iso-b0-white                   ID  ::= { id-vc-medium  220 }
id-val-medium-iso-b1-white                   ID  ::= { id-vc-medium  221 }
id-val-medium-iso-b2-white                   ID  ::= { id-vc-medium  222 }
id-val-medium-iso-b3-white                   ID  ::= { id-vc-medium  224 }
id-val-medium-iso-b6-white                   ID  ::= { id-vc-medium  226 }
id-val-medium-iso-b7-white                   ID  ::= { id-vc-medium  227 }
id-val-medium-iso-b8-white                   ID  ::= { id-vc-medium  228 }
id-val-medium-iso-b9-white                   ID  ::= { id-vc-medium  229 }
id-val-medium-iso-b10-white                  ID  ::= { id-vc-medium  230 }
id-val-medium-jis-b0-white                   ID  ::= { id-vc-medium  240 }
id-val-medium-jis-b1-white                   ID  ::= { id-vc-medium  241 }
id-val-medium-jis-b2-white                   ID  ::= { id-vc-medium  242 }
id-val-medium-jis-b3-white                   ID  ::= { id-vc-medium  243 }
id-val-medium-jis-b6-white                   ID  ::= { id-vc-medium  246 }
id-val-medium-jis-b7-white                   ID  ::= { id-vc-medium  247 }
id-val-medium-jis-b8-white                   ID  ::= { id-vc-medium  248 }
id-val-medium-jis-b9-white                   ID  ::= { id-vc-medium  249 }
id-val-medium-jis-b10-white                  ID  ::= { id-vc-medium  250 }
id-val-medium-a                              ID  ::= { id-vc-medium  260 }
id-val-medium-b                              ID  ::= { id-vc-medium  261 }
id-val-medium-c                              ID  ::= { id-vc-medium  262 }
id-val-medium-d                              ID  ::= { id-vc-medium  263 }
id-val-medium-e                              ID  ::= { id-vc-medium  264 }
-- Medium Type Object Identifier Values
id-val-medium-type-stationery                ID  ::= { id-vc-medium-type  0 }
id-val-medium-type-transparency              ID  ::= { id-vc-medium-type  1 }
id-val-medium-type-envelope                  ID  ::= { id-vc-medium-type  2 }
id-val-medium-type-envelope-plain            ID  ::= { id-vc-medium-type  3 }
id-val-medium-type-envelope-window           ID  ::= { id-vc-medium-type  9 }
id-val-medium-type-continuous-long           ID  ::= { id-vc-medium-type  4 }
id-val-medium-type-continuous-short          ID  ::= { id-vc-medium-type  5 }
id-val-medium-type-tab-stock                 ID  ::= { id-vc-medium-type  6 }
id-val-medium-type-multi-part-form           ID  ::= { id-vc-medium-type  7 }
id-val-medium-type-labels                    ID  ::= { id-vc-medium-type  8 }
id-val-medium-type-multi-layer               ID  ::= { id-vc-medium-type  10 }
-- Medium Size Object Identifier Values
id-val-medium-size-iso-a0                    ID  ::= { id-vc-medium-size  20 }
id-val-medium-size-iso-a1                    ID  ::= { id-vc-medium-size  21 }
id-val-medium-size-iso-a2                    ID  ::= { id-vc-medium-size  22 }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642946 568


```
id-val-medium-size-iso-a3                    ID  ::= { id-vc-medium-size  23 }
id-val-medium-size-iso-a4                    ID  ::= { id-vc-medium-size  24 }
id-val-medium-size-iso-a5                    ID  ::= { id-vc-medium-size  25 }
id-val-medium-size-iso-a6                    ID  ::= { id-vc-medium-size  26 }
id-val-medium-size-iso-a7                    ID  ::= { id-vc-medium-size  27 }
id-val-medium-size-iso-a8                    ID  ::= { id-vc-medium-size  28 }
id-val-medium-size-iso-a9                    ID  ::= { id-vc-medium-size  29 }
id-val-medium-size-iso-a10                   ID  ::= { id-vc-medium-size  30 }
id-val-medium-size-iso-b0                    ID  ::= { id-vc-medium-size  40 }
id-val-medium-size-iso-b1                    ID  ::= { id-vc-medium-size  41 }
id-val-medium-size-iso-b2                    ID  ::= { id-vc-medium-size  42 }
id-val-medium-size-iso-b3                    ID  ::= { id-vc-medium-size  43 }
id-val-medium-size-iso-b4                    ID  ::= { id-vc-medium-size  44 }
id-val-medium-size-iso-b5                    ID  ::= { id-vc-medium-size  45 }
id-val-medium-size-iso-b6                    ID  ::= { id-vc-medium-size  46 }
id-val-medium-size-iso-b7                    ID  ::= { id-vc-medium-size  47 }
id-val-medium-size-iso-b8                    ID  ::= { id-vc-medium-size  48 }
id-val-medium-size-iso-b9                    ID  ::= { id-vc-medium-size  49 }
id-val-medium-size-iso-b10                   ID  ::= { id-vc-medium-size  50 }
id-val-medium-size-iso-c3                    ID  ::= { id-vc-medium-size  63 }
id-val-medium-size-iso-c4                    ID  ::= { id-vc-medium-size  64 }
id-val-medium-size-iso-c5                    ID  ::= { id-vc-medium-size  65 }
id-val-medium-size-iso-c6                    ID  ::= { id-vc-medium-size  66 }
id-val-medium-size-iso-designated-long       ID  ::= { id-vc-medium-size  67 }
id-val-medium-size-jis-b0                    ID  ::= { id-vc-medium-size  80 }
id-val-medium-size-jis-b1                    ID  ::= { id-vc-medium-size  81 }
id-val-medium-size-jis-b2                    ID  ::= { id-vc-medium-size  82 }
id-val-medium-size-jis-b3                    ID  ::= { id-vc-medium-size  83 }
id-val-medium-size-jis-b4                    ID  ::= { id-vc-medium-size  84 }
id-val-medium-size-jis-b5                    ID  ::= { id-vc-medium-size  85 }
id-val-medium-size-jis-b6                    ID  ::= { id-vc-medium-size  86 }
id-val-medium-size-jis-b7                    ID  ::= { id-vc-medium-size  87 }
id-val-medium-size-jis-b8                    ID  ::= { id-vc-medium-size  88 }
id-val-medium-size-jis-b9                    ID  ::= { id-vc-medium-size  89 }
id-val-medium-size-jis-b10                   ID  ::= { id-vc-medium-size  90 }
id-val-medium-size-na-letter                 ID  ::= { id-vc-medium-size  0 }
id-val-medium-size-na-legal                  ID  ::= { id-vc-medium-size  1 }
id-val-medium-size-na-10x13-envelope         ID  ::= { id-vc-medium-size  2 }
id-val-medium-size-na-9x12-envelope          ID  ::= { id-vc-medium-size  3 }
id-val-medium-size-na-number-10-envelope     ID  ::= { id-vc-medium-size  4 }
id-val-medium-size-na-7x9-envelope           ID  ::= { id-vc-medium-size  5 }
id-val-medium-size-na-9x11-envelope          ID  ::= { id-vc-medium-size  6 }
id-val-medium-size-na-10x14-envelope         ID  ::= { id-vc-medium-size  7 }
id-val-medium-size-na-number-9-envelope      ID  ::= { id-vc-medium-size  8 }
id-val-medium-size-na-6x9-envelope           ID  ::= { id-vc-medium-size  9 }
id-val-medium-size-na-10x15-envelope         ID  ::= { id-vc-medium-size  10 }
id-val-medium-size-a                         ID  ::= { id-vc-medium-size  11 }
id-val-medium-size-b                         ID  ::= { id-vc-medium-size  12 }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642947 4T4


```
id-val-medium-size-c                        ID  ::= { id-vc-medium-size  13 }
id-val-medium-size-d                        ID  ::= { id-vc-medium-size  14 }
id-val-medium-size-e                        ID  ::= { id-vc-medium-size  15 }
-- Medium Color Object Identifier Values
id-val-medium-colour-white                  ID  ::= { id-vc-medium-colour  0 }
id-val-medium-colour-pink                   ID  ::= { id-vc-medium-colour  1 }
id-val-medium-colour-yellow                 ID  ::= { id-vc-medium-colour  2 }
id-val-medium-colour-buff                   ID  ::= { id-vc-medium-colour  3 }
id-val-medium-colour-goldenrod              ID  ::= { id-vc-medium-colour  4 }
id-val-medium-colour-blue                   ID  ::= { id-vc-medium-colour  5 }
id-val-medium-colour-green                  ID  ::= { id-vc-medium-colour  6 }
id-val-medium-colour-transparent-untinted   ID  ::= { id-vc-medium-colour  7 }
-- Medium Tooth Object Identifier Values
id-val-medium-tooth-fine                    ID  ::= { id-vc-medium-tooth  1 }
id-val-medium-tooth-coarse                  ID  ::= { id-vc-medium-tooth  2 }
-- Medium Grain Object Identifier Values
id-val-medium-grain-long                    ID  ::= { id-vc-medium-grain  1 }
id-val-medium-grain-short                   ID  ::= { id-vc-medium-grain  2 }
-- Auxilliary Sheet Object Identifier Values
id-val-auxiliary-sheet-start-sheet-brief    ID  ::= { id-vc-auxiliary-sheet  1 }
id-val-auxiliary-sheet-start-sheet-full     ID  ::= { id-vc-auxiliary-sheet  2 }
id-val-auxiliary-sheet-separator-sheet-brief ID  ::= { id-vc-auxiliary-sheet  4 }
id-val-auxiliary-sheet-separator-sheet-full  ID  ::= {id-vc-auxiliary-sheet   5 }
id-val-auxiliary-sheet-end-sheet-brief      ID  ::= { id-vc-auxiliary-sheet  6 }
id-val-auxiliary-sheet-end-sheet-full       ID  ::= { id-vc-auxiliary-sheet  7 }
id-val-auxiliary-sheet-blank                ID  ::= { id-vc-auxiliary-sheet  3 }
id-val-auxiliary-sheet-slipsheet            ID  ::= { id-vc-auxiliary-sheet  8 }
-- Stitching Object Identifier Values
id-val-stitching-default                    ID  ::= { id-vc-stitching  0 }
id-val-stitching-staple                     ID  ::= { id-vc-stitching  1 }
id-val-stitching-staple-top-left            ID  ::= { id-vc-stitching  2 }
id-val-stitching-staple-bottom-left         ID  ::= { id-vc-stitching  3 }
id-val-stitching-staple-top-right           ID  ::= { id-vc-stitching  4 }
id-val-stitching-staple-bottom-right        ID  ::= { id-vc-stitching  5 }
id-val-stitching-saddle-stitch              ID  ::= { id-vc-stitching  6 }
id-val-stitching-edge-stitch                ID  ::= { id-vc-stitching  7 }
id-val-stitching-edge-stitch-right          ID  ::= { id-vc-stitching  8 }
id-val-stitching-edge-stitch-top            ID  ::= { id-vc-stitching  9 }
-- Stitching Type Object Identifier Values
id-val-stitching-type-default               ID  ::= { id-vc-stitching-type  0 }
id-val-stitching-type-staple                ID  ::= { id-vc-stitching-type  3 }
id-val-stitching-type-stitch                ID  ::= { id-vc-stitching-type  4 }
id-val-stitching-type-continuous            ID  ::= { id-vc-stitching-type  5 }
-- Binding Object Identifier Values
id-val-binding-default                      ID  ::= { id-vc-binding  0 }
-- Binding Type Object Identifier Values
id-val-binding-type-default                 ID  ::= { id-vc-binding-type  0 }
id-val-binding-type-tape                    ID  ::= { id-vc-binding-type  1 }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642948 330


```
id-val-binding-type-plastic                 ID  ::= { id-vc-binding-type  2 }
id-val-binding-type-velo                    ID  ::= { id-vc-binding-type  3 }
id-val-binding-type-perfect                 ID  ::= { id-vc-binding-type  4 }
id-val-binding-type-spiral                  ID  ::= { id-vc-binding-type  5 }
-- Binding Color Object Identifier Values
id-val-binding-colour-default               ID  ::= { id-vc-binding-colour  0 }
id-val-binding-colour-black                 ID  ::= { id-vc-binding-colour  1 }
id-val-binding-colour-blue                  ID  ::= { id-vc-binding-colour  2 }
id-val-binding-colour-gray                  ID  ::= { id-vc-binding-colour  3 }
id-val-binding-colour-brown                 ID  ::= { id-vc-binding-colour  4 }
-- Perforation Object Identifier Values
id-val-perforation-default                  ID  ::= { id-vc-perforation  0 }
-- Perforation Type Object Identifier Values
id-val-perforation-type-default             ID  ::= { id-vc-perforation-type
0 }
-- Trimming Object Identifier Values
id-val-trimming-default                     ID  ::= { id-vc-trimming  0 }
-- Punching Object Identifier Values
id-val-punching-default                     ID  ::= { id-vc-punching  0 }
-- Slitting Object Identifier Values
id-val-slitting-default                     ID  ::= { id-vc-slitting  0 }
-- Covers Object Identifier Values
id-val-covers-default                       ID  ::= { id-vc-covers  0 }
-- Cover Name Object Identifier Values
id-val-cover-name-default                   ID  ::= { id-vc-cover-name  0 }
-- Folding Object Identifier Values
id-val-folding-default                      ID  ::= { id-vc-folding  0 }
-- Plex Object Identifier Values
id-val-plex-simplex                         ID  ::= { id-vc-plex  0 }
id-val-plex-duplex                          ID  ::= { id-vc-plex  1 }
id-val-plex-tumble                          ID  ::= { id-vc-plex  2 }
-- Stacking Object Identifier Values
id-val-output-stacking-default              ID  ::= { id-vc-output  10 }
END
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4851903 0642949 277

© ISO/IEC

Annex G
(informative)

Bibliography

The following standards or working drafts of standards contain information which, through reference in this text, supplement or explain the provisions of this International Standard, or which may become normative references at a future time

ISO/IEC 8613-1:1994, *Information technology — Open Document Architecture (ODA) and interchange format: Introduction and general principles.*

ISO/IEC 8613-2:1995, *Information technology — Open Document Architecture (ODA) and interchange format: Document structures.*

ISO/IEC 8613-4:1994, *Information technology — Open Document Architecture (ODA) and interchange format: Document profile.*

ISO/IEC 8613-5:1994, *Information technology — Open Document Architecture (ODA) and interchange format: Document interchange format.*

ISO/IEC 8613-6:1994, *Information technology — Open Document Architecture (ODA) and interchange format: Character content architectures.*

ISO/IEC 8613-7:1994, *Information technology — Open Document Architecture (ODA) and interchange format: Raster graphics content architectures.*

ISO/IEC 8613-8:1994, *Information technology — Open Document Architecture (ODA) and interchange format: Geometric graphics content architectures.*

ISO/IEC 10179:—[3], *Information technology — Text and office systems — Document Style Semantics and Specification Language (DSSSL).*

TIFF 6.0 Specification, Aldus Corporation, Seattle, WA June 1992.

3) To be published.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ICS 35.240.30

Descriptors: data processing, information interchange, text communication, documentation, documents, electronic documents, text processing, artificial languages.

Price based on 464 pages

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale