4851903 0607735 874

INTERNATIONAL STANDARD

ISO/IEC 12087-1

First edition
1995-04-15

Information technology — Computer graphics and image processing — Image Processing and Interchange (IPI) — Functional specification —

Part 1:
Common architecture for imaging

Technologies de l'information — Infographie et traitement de l'image — Traitement de l'image et échange (IPI) — Spécification fonctionnelle —

Partie 1: Architecture commune pour l'image

Reference number
ISO/IEC 12087-1:1995(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

4851903 0607736 700

Contents

1 Scope 1

2 Normative References 3

3 Definitions and abbreviations 4
3.1 Definitions 4
3.2 Abbreviations 5
3.3 Diagrammatic Conventions 5

4 The IPI architecture 7
4.1 IPI imaging architecture 7
4.1.1 IPI imaging model 7
4.1.2 IPI operator processing model 8
4.2 IPI basic data types 9
4.2.1 IPI elementary data types 10
4.2.2 IPI compound data types 10
4.3 IPI image data types 11
4.3.1 IPI derived elementary image data types 11
4.3.2 IPI derived compound image data types 11
4.3.3 IPI derived image attributes 13
4.4 IPI derived non-image data types 14
4.4.1 IPI derived image annotation data types 14
4.4.2 IPI derived image-related non-image data types 14

5 IPI-PIKS architecture 16
5.1 IPI-PIKS imaging model 16
5.1.1 IPI-PIKS neighbourhood control 16
5.1.2 IPI-PIKS image control 17
5.2 IPI-PIKS system control 17
5.2.1 Data object management 17
5.2.2 Operational synchronicity 18
5.2.3 Element chaining 18
5.2.4 Error management 18
5.3 IPI-PIKS basic data types 18
5.3.1 IPI-PIKS elementary data types 18
5.3.2 IPI-PIKS compound data types 19
5.4 IPI-PIKS derived image data descriptions 22
5.4.1 IPI-PIKS derived data types 22
5.4.2 IPI-PIKS compound image data types 22
5.4.3 Composite images 23
5.4.4 IPI-PIKS image object attributes 24
5.5 IPI-PIKS derived non-image data structures 26
5.6 IPI-PIKS data pragmata 34

© 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 0607737 647

6 IPI-IIF-specific architecture . . . 35
6.1 IPI-IIF imaging model . . . 35
6.2 IPI-IIF basic data types . . . 35
6.3 IPI-IIF derived data types . . . 36
6.3.1 IPI-IIF derived image data types . . . 36
6.3.2 IPI-IIF image attributes . . . 36
6.3.3 IPI-IIF derived non-image data types . . . 37
6.3.3.1 IPI-IIF image annotation data types . . . 37
6.3.3.2 IPI-IIF image-related non-image data types . . . 37

7 Relationship between IPI-PIKS and IPI-IIF . . . 41

8 Conformance . . . 42
8.1 Conformance of functionality . . . 42
8.2 Conformance of accuracy and precision . . . 42
8.3 Extensions . . . 42
8.4 Conformance profiles . . . 43
8.4.1 Types of profile . . . 43
8.4.2 Application profile registration . . . 44
8.4.3 Profiles defined by IPI . . . 44

Annexes . . . 46

A Structured image data types . . . 46

B Structure codes . . . 48

C The representation of colour . . . 49

D Language-Independent Data Types . . . 56
D.1 Bit . . . 56
D.2 Boolean . . . 56
D.3 Character . . . 57
D.4 Complex . . . 58
D.5 Enumerated . . . 58
D.6 Null . . . 59
D.7 Integer . . . 59
D.8 Real . . . 60
D.9 State . . . 61
D.10 Array . . . 61
D.11 Choice . . . 62
D.12 List . . . 63
D.13 Pointer . . . 64
D.14 Range . . . 65
D.15 Record . . . 65
D.16 Set . . . 66
D.17 Character String . . . 67
D.18 Table . . . 68

E Bibliography . . . 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 0607738 583

List of figures

1 Relationship of the parts of ISO/IEC 12087 2
2 Diagrammatic conventions 6
3 Interfaces between application program, IPI-PIKS, and IPI-IIF 8
4 Fundamental operator processing model 9
5 The operator model used by IPI-PIKS 16
6 Relationship Between a Physical Volume and IPI-PIKS Horizontal, Vertical, and Depth Coordinates 23
7 Aggregation of Image References into a List 24
8 Colour Systems and Representations Used by IPI 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 0607739 41T

List of tables

1 Codes for the externally-visible representations of IPI-PIKS-specific data types 21
2 Dimensions of an IPI-PIKS Data Object 22
3 IPI-IIF profiles that correspond to IPI-PIKS profiles 44
4 IPI-PIKS profiles that correspond to IPI-IIF profiles 45
5 XYZ tristimulus values for the white points of common illuminance 51
6 Supported types of colour representation, and their attributes 53
7 Standardized parameterisations of colours 53
8 Parameter values for the standardized colour representations (non-normative) 54
9 Mappings Between Colours and Image Channels 55

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

4851903 0607740 131

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 12087-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 24, *Computer graphics and image processing*.

ISO/IEC 12087 consists of the following parts, under the general title *Information technology — Computer graphics and image processing — Image processing and interchange (IPI) — Functional specification*:

— *Part 1: Common architecture for imaging*

— *Part 2: Programmer's imaging kernel system application programme interface*

— *Part 3: Image Interchange Facility (IIF)*

Annexes A to D form an integral part of this part of ISO/IEC 12087. Annex E is 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 0607741 078

Introduction

The processing of images is a requirement of many application areas of information processing. Early work in these areas led to the development of many application program interfaces and a large number of image representations for interchange. The purpose of ISO/IEC 12087 is to provide an application program interface and an image interchange representation in order to increase the portability of application software.

ISO/IEC 12087 provides an architectural model for the representation and manipulation of images in a digital form. Based on this model, it defines an application program interface and an image interchange format. It is applicable to all application areas that involve the processing, manipulation, or transfer of image data.

ISO/IEC 12087 includes notes and exemplary material. Such material is non-normative: it is included solely to aid understanding and does not form part of ISO/IEC 12087.

ISO/IEC 12087 initially comprises three parts:

1 *Common architecture for imaging*, which describes the common architectural material on which the entire Standard is based;

2 *Programmer's imaging kernel system application program interface*, which defines processing operations to be carried out on image data;

3 *Image Interchange Facility (IIF)*, which defines how images may be interchanged between application programs.

Information may be interchanged between the application program, Programmer's Imaging Kernel System (IPI-PIKS), and Image Interchange Facility (IPI-IIF) (see figure). Data paths between all three components are standardized in ISO/IEC 12087, as indicated by the solid lines; however, it is also permitted that implementations may use private, implementation-dependent data paths, shown by dashed lines; such data paths are outside the scope of ISO/IEC 12087.

There are a great many types of application that involve the use of images. The Computer Graphics Reference Model [ISO 11072] identifies six main function classes (see figure 0.1):

image analysis — transformation of digital images to image and non-image data; this encompasses basic functions such as histogram generation, mean value determination, image classification, *etc.*, but does not include image understanding using artificial intelligence techniques.

image interpretation — the process of inferring symbolic scene descriptions from image data.

image presentation — transformation of image data to a form suitable for an observer; *e.g.*, via video monitors, printers, film recorders, *etc.*

image processing — transformation of digital images to digital images; *e.g.*, grey value contrast enhancement, edge detection, *etc.*

image sensing — transformation of real-world information to digital images; *e.g.*, via cameras, optical scanners, *etc.*

image synthesis — transformation of non-image data to image data; this encompasses functions such as the rendering of lines, creation of test images, simulation of sensor functions, letters of graphical text and symbols, *etc.*

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

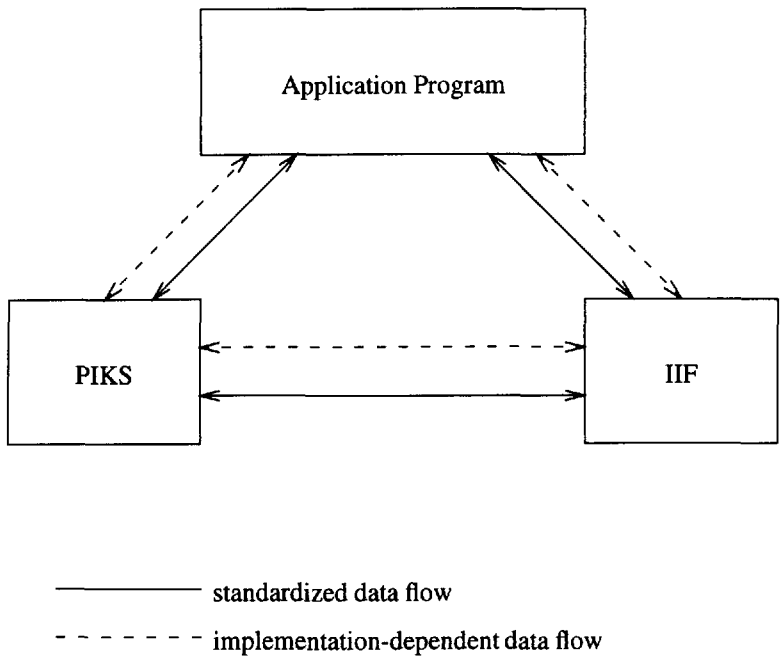


Figure 0.1 — Data flow between the application program, IPI-PIKS, and IPI-IIF

As figure indicates, all these function classes involve the manipulation of a digital image; some function classes also require information that is related to the data contained in the digital image but is itself non-image in nature. This *image-related* information is essential to many of the common operations performed on digital images and is therefore also described by ISO/IEC 12087.

ISO/IEC 12087 is also concerned with *image interchange,* the interchange of digital images among imaging applications; this serves for the communication of image data and related non-image data among imaging applications.

The term 'digital image' used in [ISO 11072] is synonymous with the term 'image' as used in ISO/IEC 12087. It is important to realize the distinction between 'image' (or 'digital image') as used in ISO/IEC 12087 and the term 'image' as it may be used colloquially: in ISO/IEC 12087, 'image' (or 'digital image') refers to a particular representation of image data within a computer system. An image may not be viewed directly. To view an image, an explicit presentation step is involved, as figure indicates. Image data that are in a form suitable for viewing by an observer are termed 'presentable' image data in ISO/IEC 12087.

NOTE 1 Some application areas, which might loosely be termed "image understanding," utilize data derived from an image by means of some analysis; such applications are therefore omitted from this ISO/IEC 12087. However, ISO/IEC 12087 may be used by such applications.

This part of ISO/IEC 12087 fulfills the following purposes:

a) It provides an overview of ISO/IEC 12087;

b) It defines a Common Architecture for Imaging, an abstract architectural model for the representation

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

4851903 0607743 940

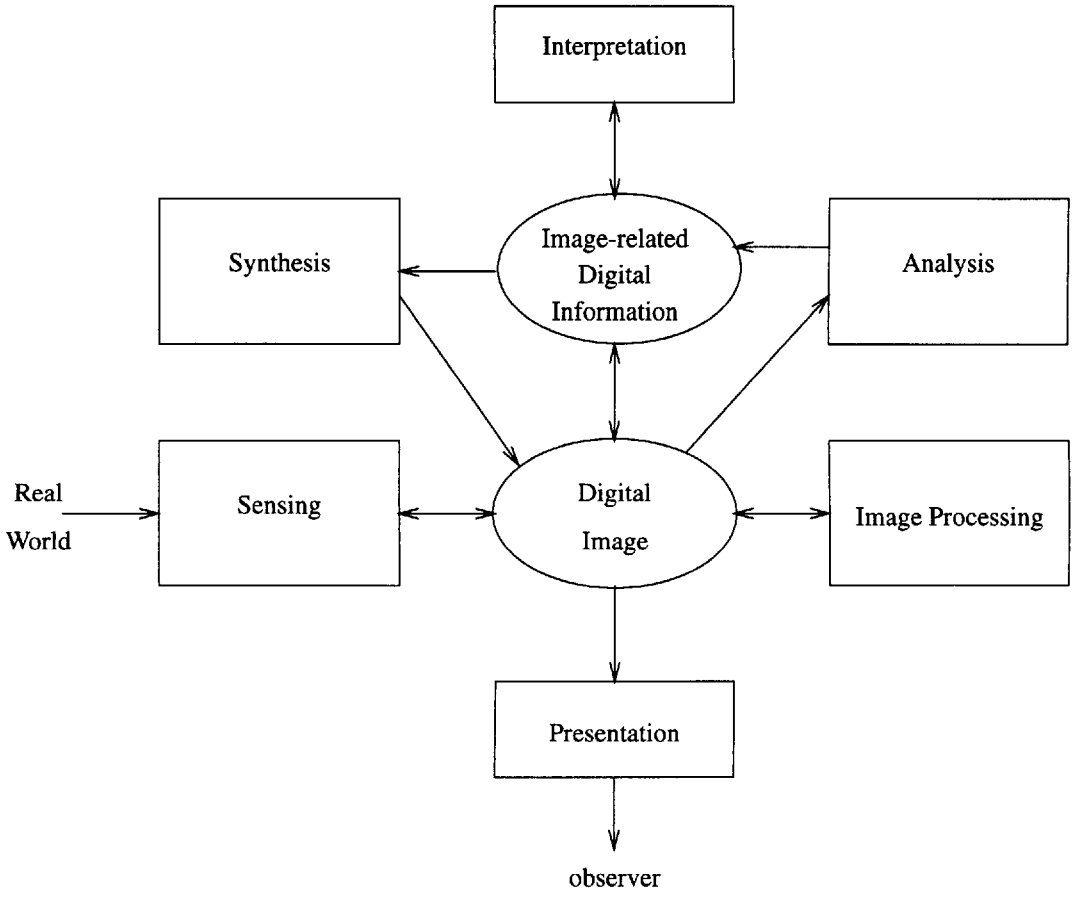


Figure 0.2 — Classes of operations on images

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

4851903 0607744 887

and processing of image data. The purpose of this model is to define a common set of data types and a common image representation for use with all other parts of ISO/IEC 12087 and to provide a standardized framework upon which future imaging standards may be built, allowing simplified conversion of existing applications to the new standard.

c) It defines rules to which conforming implementations shall adhere and the mechanism by which conformance is 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 0607745 713

Information technology — Computer graphics and image processing — Image Processing and Interchange (IPI) — Functional specification —

Part 1:

Common architecture for imaging

1 Scope

ISO/IEC 12087 is concerned with the manipulation, processing, and interchange of all types of digital images. The main purpose of this part is to define a generic, unifying imaging architecture to which other parts of ISO/IEC 12087 conform. This part of ISO/IEC 12087 also defines those "specializations" or "delineations" of the generic imaging architecture that are required to support IPI-PIKS and IPI-IIF.

The relationship of the different parts of ISO/IEC 12087 is shown in figure 1. This part of ISO/IEC 12087 describes material that applies throughout ISO/IEC 12087, including topics such as data types available for use in image data and image-related data, and a model for the processing of digital images by operators. These topics are presented in a general form, since it is intended that subsequent imaging standards will conform to the same architectural model.

Derived from this general description are more constrained descriptions of the same topics. The principal reason for this process of delineation is to restrict the range of data representations for IPI-PIKS and IPI-IIF, while simultaneously ensuring that IPI-IIF is capable of interchanging both IPI-PIKS data objects and objects that cannot be represented or manipulated within IPI-PIKS.

ISO/IEC 12087 permits multiple Application Program Interface (API)s to be developed, each of which must be

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

4851903 0607746 65T

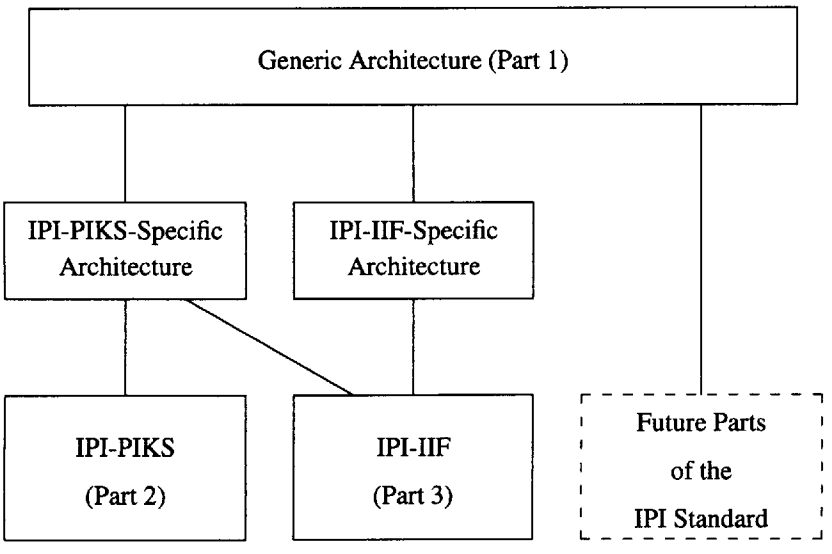


Figure 1 — Relationship of the parts of ISO/IEC 12087

based on specific delineations of the imaging model described herein. Each API will be specified in a separate part of ISO/IEC 12087. Any subsequent APIs developed as part of ISO/IEC 12087 must conform to the common architecture described in this document, and must be extensions of the APIs described in ISO/IEC 12087-2 and ISO/IEC 12087-3.

ISO/IEC 12087 is intended for use in a wide variety of environments where digital images are handled.

NOTE 2 Application areas that are addressed by Image Processing and Interchange (IPI) include: image manipulation; image enhancement; image analysis; and image transport. Application areas that are not addressed by IPI include: computer graphics; image understanding; multimedia; device control; and window systems.

ISO/IEC 12087 is intended to conform with other International Standards developed to handle digital images. Such standards include the JPEG [ISO/IEC 10918-1:1994], and MPEG [ISO/IEC 11172-1:1993] compression standards, Open Systems Interconnect [ISO/IEC 8824:1990], and Office Document Architecture [ISO/IEC 8613]. Those aspects of ISO/IEC 12087 that are concerned with the acquisition and display of digital images conform with the Computer Graphics Reference Model [ISO 11072]. Furthermore, annex B of [ISO 11072] describes how imaging fits within the general framework of that model. ISO/IEC 12087-3 uses Abstract Syntax Notation 1 [ISO/IEC 8824:1990] in the definition of the image interchange format.

ISO/IEC 12087 complies directly with all standards listed in clause 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 0607747 596

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this part of the IPI Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on ISO/IEC 12087 are encouraged to investigate the possibility of applying the most recent editions of the standards listed 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 8613:1989, *Information processing — Text and office systems — Office Document Architecture (ODA) and interchange format.*

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/IEC 9973:1994, *Information technology, Computer graphics and image processing — Procedures for registration of graphical items.*

ISO/IEC 10918-1:1994, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines.*

ISO/IEC 11072:1992, *Information technology — Computer graphics — Computer Graphics Reference Model.*

ISO/IEC 11172-1:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s — Part 1: Systems.*

CIE:1931, *Proceedings of the eighth session, Cambridge, England, 1931.* Bureau Centrale de la CIE, Paris.

CIE:1970, *International Lighting Vocabulary.* CIE publication no. 17, third edition, Bureau Centrale de la CIE.

CIE:1976, *CIE recommendations on uniform colour space, colour difference equations, psychomatic colour terms.* Supplement no. 2 to CIE publication 15, Bureau Centrale de la CIE, Paris.

CCIR:1990a, *Report 476-1, Colorimetric standards in colour television.* Recommendations of the CCIR 1990, Annex to Volume XI – Part 1 – Broadcasting Service (Television), CCIR, Geneva.

CCIR:1990b, *Report 624-4, Characterization of television systems.* Recommendation of the CCIR 1990, Annex to Volume XI – Part 1 – Broadcasting Service (Television), CCIR, Geneva.

CCIR:1990c, *Recommendation 709, Basic parameter values of the HDTV standard for the studio and for international programme exchange.* Recommendations of the CCIR 1990, Volume XI – Part 1 – Broadcasting Service (Television), CCIR, Geneva.

EBU:1975, *EBU standard for chromaticity tolerances for studio monitors.* Tech. 3213-E, Brussels.

DeMarsh:1974, *Colorimetric standards in U. S. color television. A report by the subcommittee on systems colorimetry of the SMPTE television committee.* L. E. DeMarsh, Journal of the Society of Motion Picture and Television Engineers, vol. 83.

NTSC:1954, *NTSC signal specifications.* Proceedings of the IRE, January 1954.

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

4851903 0607748 422

3 Definitions and abbreviations

3.1 Definitions

For the purpose of ISO/IEC 12087, the following definitions apply.

3.1.1 digital image: Synonymous with image.

3.1.2 element: A collective noun used to describe mechanism, operator, tool, and utility.

3.1.3 image: A data structure that contains pixels and image-related data.

3.1.4 image data: Digital values corresponding to pixels.

3.1.5 image-related data: Information that provides context for the interpretation of an image.

3.1.6 mechanism: A software element that performs control and management tasks.

3.1.7 operator: A software element that performs manipulation of an image or set of images. Permitted operator types are image-to-image, image-to-non-image, and non-image-to-non-image.

NOTE 3 Operators may be based upon other operators — in which case they are said to be high-level operators — or may not, in which case they are said to be primitive. All classes of IPI-PIKS operators, tools, and utilities may be invoked from an application program, but only operators and utilities interface directly to the underlying system.

3.1.8 primitive operator: One of a small set of low-level operators that is fundamental to image processing.

3.1.9 pixel: An abbreviation for the term 'picture element.'

NOTE 4 In some application areas, the term 'pixel' commonly refers to 2D images only, with separate terms (*e.g.*, 'voxel') being used to describe an element of a higher-dimensional image. In this Standard, the single term 'pixel' is used to refer an element of an image of any dimensionality.

3.1.10 picture element: A digital representation of one or more quantities at a particular location within an image. A picture element may have an associated physical metric, size, or interval.

3.1.11 tool: A software element that creates data objects to be used by operators. Examples of tools are test image generation and the generation of mathematical functions.

3.1.12 utility: A software element that performs basic image manipulation operations (*e.g.*, image import, access, and memory management).

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

4851903 0607749 369

3.2 Abbreviations

API	Application Program Interface
CAI	Common Architecture for Imaging
CCIR	Comité Consultatif International des Radiocommunications
CGI	Computer Graphics Interface
CGM	Computer Graphics Metafile
CIE	Commission Internationale d'Eclairage
CMY	Cyan, Magenta, Yellow (colour model)
CMYK	Cyan, Magenta, Yellow, Black (colour model)
EBU	European Broadcasting Union
GKS	Graphical Kernel System
JPEG	Joint Photographic Experts Group
IHS	Intensity, Hue, Saturation (colour representation)
IIF	Image Interchange Facility
MPEG	Motion Picture Experts Group
NMR	Nuclear Magnetic Resonance
NTSC	National Television Standards Committee
ODA	Office Document Architecture
ODIF	Office Document Interchange Format
PET	Positron Emission Tomography
PHIGS	Programmer's Hierarchical Interactive Graphics System
PIKS	Programmer's Imaging Kernel System
RGB	Red, Green, Blue (colour representation)
ROI	region of interest (defined in clause 4.4.2)
SMPTE	Society of Motion Picture and Television Engineers

3.3 Diagrammatic Conventions

Diagrams are used throughout ISO/IEC 12087 to support textual descriptions. The graphical items that comprise these diagrams have a consistent meaning, as shown in figure 2, unless indicated to the contrary.

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

4851903 0607750 080

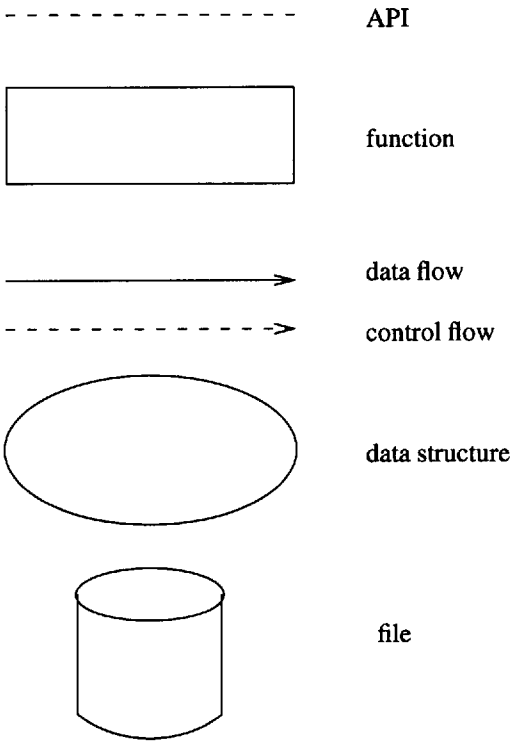


Figure 2 — Diagrammatic conventions

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

4851903 0607751 T17

4 The IPI architecture

4.1 IPI imaging architecture

In general, imaging systems deal with several types of data falling in the general categories of image data and non-image data, all of which are normally related to physical phenomena. Examples of these categories are:

image data — grey-scale images, bi-tonal images, colour images, multichannel satellite data, image sequences, sequences of stereo pair images, *etc.*;

non-image data — look-up tables, histograms, text, graphics, regions of interest, *etc.*;

real world information — continuous physical variables such as light intensity and wavelength, pressure, temperature, continuous chemical variables, *etc.*

For the purpose of ISO/IEC 12087, three categories of data types are defined:

basic data types — these are the data types that must be supported by the system upon which ISO/IEC 12087 is implemented; the category includes both 'elementary' data types (*e.g.*, Boolean, integer) and 'derived' (structured) data types (*e.g.*, array, record).

image data types — these are derived data types that represent image data for the purposes of processing and interchange.

non-image data types — these are derived data types that represent image-related, but non-image, data (*e.g.*, region of interest).

These data types are defined in general terms in this clause. Note that data types are regarded as identical if they are of identical structure with respect to the categories defined in this clause.

4.1.1 IPI imaging model

Figure 3 shows the relationship between the parts of an IPI implementation. In particular,

- IPI-PIKS and IPI-IIF are self-contained parts of ISO/IEC 12087, and may be used either independently or in conjunction by an application program;
- an application program may apply imaging functions via IPI-PIKS;
- an application program may import and export images directly via IPI-IIF;
- any interpretation of data types other than those of the three abovementioned categories must be performed by the application program;
- an application program controls how and to what extent IPI-PIKS and IPI-IIF are used;
- an application program may use other standards for information processing (*e.g.*, GKS, CGI, PHIGS) or for data interchange (*e.g.*, CGM, ODA/ODIF);

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

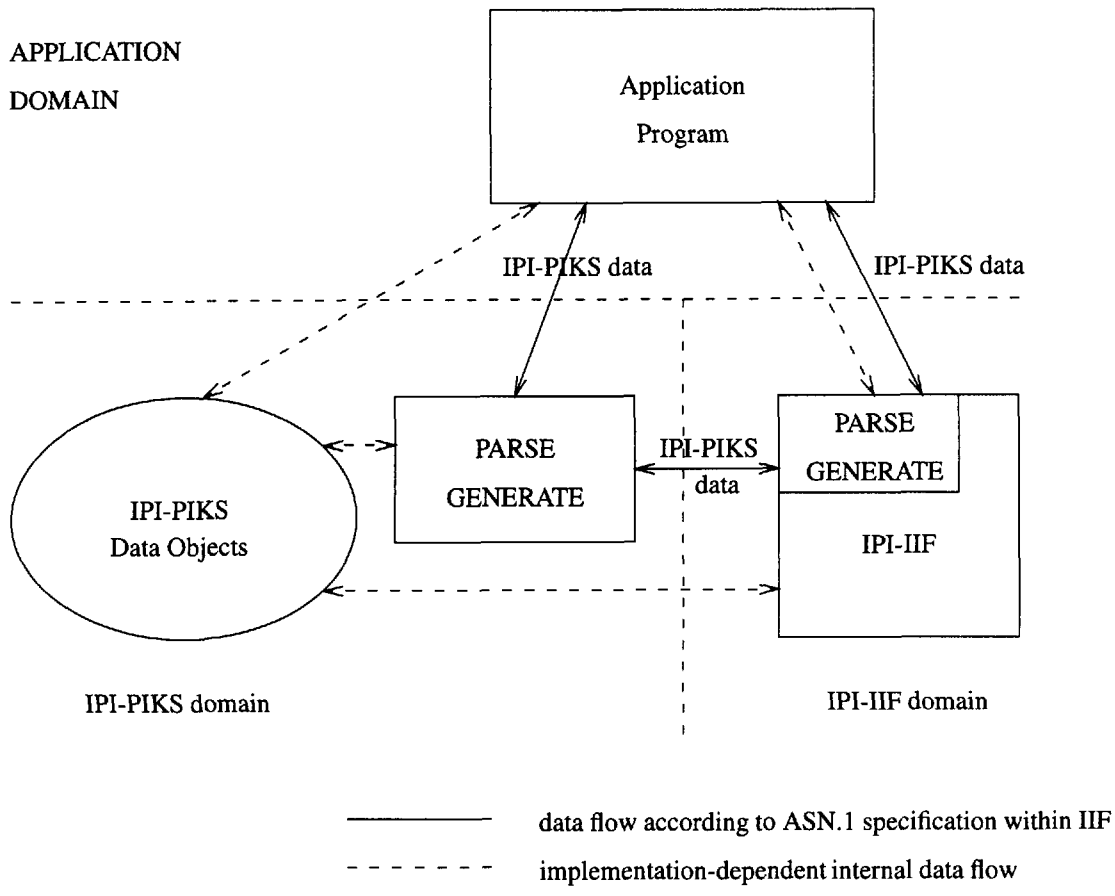


Figure 3 — Interfaces between application program, IPI-PIKS, and IPI-IIF

— an application program may perform user interaction as well as the display and presentation of images to the user;

— an application program is responsible for the consistent use of different standards.

Note that the parsing and generation of an IPI-IIF data stream is capable of being performed both within IPI-IIF itself, or in IPI-PIKS when importing or exporting images from an IPI-IIF data stream. The parsing and generation capability within IPI-PIKS is a subset of that available in IPI-IIF. See clause 7 for further details.

4.1.2 IPI operator processing model

The fundamental operator model for IPI contains three elements, which are summarized in figure 4. Note that this model does not apply to IPI-PIKS Tools, Utilities, and Mechanisms. The three elements are:

an operator — which performs one of three possible conversions:

— source image data to destination image data;

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

4851903 0607753 89T

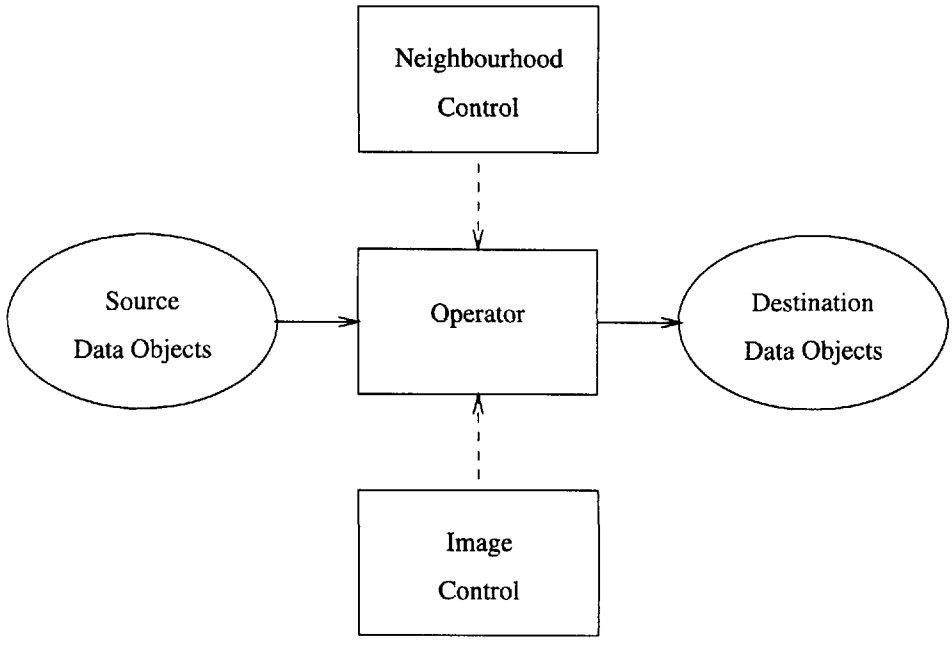


Figure 4 — Fundamental operator processing model

— source image data to destination non-image data;

— source non-image data to destination non-image data.

an image control element — which selects source image pixels to be processed, controls processing options, and selects which processed source pixels are to be recorded in the destination image.

a neighbourhood control element — a means of describing a region around a pixel, not bound to any one pixel, that identifies source image pixels to be processed and selects processing options, subject to the image control.

NOTE 5 The image control and neighbourhood control elements serve distinct purposes. Image control is used to identify that region of the image within which processing will occur: it is necessarily bound to the image. An example of this is the region of interest, discussed in this clause and elsewhere. However, neighbourhood control is used to specify a small region which is not bound to the image; an example of this is an impulse response array used by convolution operators.

4.2 IPI basic data types

For the purpose of ISO/IEC 12087, two categories of basic data types exist: elementary data types and compound data types. Compound data types are constructed from elementary data types via construction mechanisms (see clause 4.2.2). Other data types, such as image data types and non-image data types, shall refer to basic data types.

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

4851903 0607754 726

The basic data types directly comply with [1], which defines both simple and structured data types by the properties that they exhibit. Detailed information regarding the definitions of the basic data types and the basic operations that are supported are defined in [1]. Data types that are required by IPI but not defined in [1] are fully defined below.

The lists of basic operations for the data types defined below are not intended to define strictly minimal and complete sets of operations in a mathematical sense, but rather indicate important basic operations that are necessary to cover practical applications. All the operations listed below are not necessarily available in IPI-PIKS; they are included to limit any future APIs.

4.2.1 IPI elementary data types

The elementary data types supported by IPI are as follows:

bit
Boolean
character according to [ISO/IEC 646:1991]
complex number
enumerated
null
integer number
real number
state

These types are defined in annex 8.4.3. Each of these generic elementary data types may be delineated into one or more specific data types elsewhere in ISO/IEC 12087, which should then be treated as separate data types.

4.2.2 IPI compound data types

A compound data type is a data type that is constructed by one of the following mechanisms, also defined in annex 8.4.3, from basic data types:

$N_1 \times \cdots \times N_n$ array
character string
choice
list
pointer
range
record
set
table

Images are comprised of a number of pixels. The pixel values of an image must be constructed from the basic data types listed in clause 4.2. The construction mechanisms used to combine pixels into images are the compound data types described in this clause. Similarly, all image-related and non-image data described by ISO/IEC 12087 are constructed from the same elementary data types via the same construction mechanisms. Consequently, all these 'structured' or 'constructed' data types are referred to as *derived* data types for

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

4851903 0607755 662

the purposes of ISO/IEC 12087.

4.3 IPI image data types

This clause considers the nature of the image data types that are represented and manipulated by ISO/IEC 12087. All these data types are derived from the elementary data types of clause 4.2.1 using the construction mechanisms described in clause 4.2.2. Two categories of image data type may be derived by this method:

elementary image data type — or **fundamental image data type** in which the pixel values are of an elementary data type or a compound data type whose elements are themselves of basic data type;

compound image data type — in which the construction mechanisms of clause 4.2.2 are used to combine images, which may themselves be either elementary or compound.

Both these categories are considered below.

4.3.1 IPI derived elementary image data types

An elementary image data type describes an image that is built from pixels whose values are referenced to a single spatial position. An elementary image may be constructed from elementary data types using any of the compound data types of described earlier in this clause.

Note that IPI-PIKS is capable of representing and processing only particular categories of elementary image data types internally, although its compound image aggregation facility (clause 5.4.3) permits the application program to represent a wider range of structures.

4.3.2 IPI derived compound image data types

The central idea behind compound images in ISO/IEC 12087 is to provide structured access to large sets of related images. Heterogeneous image and non-image data collections can be combined into structures of arbitrary complexity. Pixel values may also be of arbitrary complexity: they could be elementary data types for storing scalar amplitude information; but they could also be compound data types for associating more comprehensive and structured information with every pixel. Indeed, the compound image data types deliberately provide more than one way for structuring the same amount of data to optimally meet the demands of the application. Multi-spectral images, for example, depending on the access mechanisms needed by the application or the properties of the sensors, could be represented as:

— an elementary image whose pixel values are 1D arrays of integers representing the intensities of the respective channels;

— a 1D array of elementary images whose pixel values are integers representing the intensities in one of the frequency channels, if regularly-spaced frequency channels have to be accessed at random;

— a list of elementary images whose pixel values are integers representing the intensities in one of the frequency channels, if irregularly spaced frequency channels have to be accessed sequentially;

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

4851903 0607756 5T9

— a record of elementary images whose pixel values are integers representing the intensities in one of the frequency channels, if the frequency channels are of different spatial resolutions or have to be accessed by name.

In fact, a complete hierarchy of constructs can be achieved:

— individual pixels;

— $1, 2, 3, \ldots, n$-dimensional arrays of pixels;

— images comprising arrays of pixels and additional information;

— compound images of arbitrary complexity comprising different image and non-image data types.

A compound image data type is a data type constructed by one of the following construction mechanisms from image data types:

$(N_1 \times \cdots \times N_n)$ image array — an array of images of identical image data type that can be accessed individually by their indices $(i_1, \ldots, i_n)$. Example operations: equal relation, setting and retrieving of individual components with random access. Examples of image arrays are identically-structured, local high-resolution images of regularly-spaced test patterns for the inspection of printed circuit boards or integrated circuits, regular spatio-temporal sequences of slice images through a beating heart, *etc.*

image list — an ordered image set. Individual images can be accessed only via successor/predecessor relations. Example operations: equal relation, is-empty, get-head, get-tail, append, cut, get-successor, get-predecessor, iterate-through-list. Examples of the data type are collections of related images that can be ordered according to their spatial, temporal, or frequency relation, *e.g.* stacks of Computerized Tomography or Magnetic Resonance images through a tumour in a human body, time sequences of a moving heart or moving objects in industrial scenes, images obtained at different wavelengths or energies, *e.g.* multi-spectral images, scattered X-ray images, spectroscopic sequences in Magnetic Resonance Imaging, *etc.*

image record — a collection of individually-named image data types and optional non-image data types. The record components may be of different type and are accessed via their names. Example operations: equal relation, append-component, remove-component, set-component, get-component. Examples of situations in which image records would prove useful include collections of images of an object that has been imaged with different sensors at different spatial, temporal, or frequency resolutions, *e.g.* images of increasing spatial resolution that form an image pyramid, medical images of the same patient, *etc.*

image set — a collection of images of identical image data type. The images cannot be accessed individually. It is possible only to check whether or not an image belongs to a set. Example operations: equal and subset relations, is-element-of, define-set-of, is-empty, apply-to-all-elements; set operations complement, union, intersection. Examples of image sets are sets of template images representing a certain class of objects, *e.g.* automobile parts, textures *etc.*, that are only used to check whether or not an image belongs to the class.

Note that image arrays, lists, and sets are analogous to compound data types but that image records are not, since they may contain both pixel and non-pixel data. The operations that may be performed on these compound data types conform with those described in [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 0607757 435

4.3.3 IPI derived image attributes

Image attributes carry information on how to correctly interpret the associated image data; they are inherently bound to images. Hence, image attributes are relevant to both IPI-PIKS and IPI-IIF. Image attributes may be attached either to elementary images (clause 4.3.1) or to compound images (clause 4.3.2). Attributes that are not attached to pixel rasters but to higher-level structures of compound images are effective for all lower level components (*i.e.,* the inheritance of attributes is hierarchical). However, it is possible to overwrite inherited attributes via their basic operations. Example operations for image attributes are: equal relation, attach-attribute, remove-attribute, set-attribute, get-attribute.

This clause describes image attributes in a purely textual and behavioural manner; for details on the content and data types of image attributes supported by IPI-PIKS and IPI-IIF, see clause 5.4.4 and clause 6.3.2.

channel attribute — Many types of images can be thought of as being comprised of several 'channels' of data. The channel attribute provides information about the sensor models underlying these channels that is necessary for the analysis of channel information, *e.g.* the number of bits used for quantizing the original sensor signals, sensitivity and noise characteristics of the sensors, spectral behaviour of the sensors, *etc.*

For example, a medical application might combine an MR and a PET data set into a single IPI image; the channel attribute would then be used to identify the MR and PET channels and to describe the characteristics of the two devices that captured them. Similarly, colour images are generally digitized into three separate channels, the first representing the red pixels, the second the green pixels and the third the blue pixels, and the channel attribute could be used to describe this.

colour attribute — Since colour images form such an important class to the human observer, it is essential that sufficient information is carried with a colour image for it to be reproduced accurately. Hence, a *colour attribute* is associated with colour images. Note that a colour image has a single colour description, but a separate channel description for each channel. This colour description encompasses all entities dealing with the mixing of amplitude values of multiple channels in order to achieve accurate colour reproduction; this may consist of the colour representation involved (see annex 8.4.3), test colours for regions of the image, *etc.*

control attribute — The purpose of a control attribute is to convey information that may be needed for subsequent processing or presentation of an image. This may include, for example, the match-point and the identifier of a region of interest associated with an IPI-PIKS image.

free-form attribute — The internal structure and syntax of the free-form description attribute is not specified within ISO/IEC 12087. It is application-dependent, and is provided to allow application-specific descriptive information to be attached to the pixel data of an image. It is left to the application to decide on the actual way of storing and manipulating this information. No IPI-PIKS functionality is provided to manipulate the content of the free-form description; any processing of this description must be performed entirely by the application program.

One example for a free-form description which is defined in detail by an application-specific standard is the header information as defined by the ACR-NEMA standard for medical applications. It provides specific formats for describing the nature of the image, orientation of the image with respect to the human body, patient data, *etc.* Another example is the FITS format widely adopted in the observational astronomy community, which allows tables of physical quantities associated with the recording process to be transferred as part of the image description.

Note that the use of the free-form description attribute for transmitting image-related semantics within IPI-IIF communication must be based on a transmitter/receiver agreement. Otherwise, these image-related semantics will be lost for the receiver.

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

4851903 0607758 371

representation attribute — The purpose of a representation attribute is to convey information regarding the spatial nature of the imaging system that recorded the image. This will typically provide information as to the size of the image, the data types of the pixel values, *etc.*, but it may also include details such as the sampling raster involved (*e.g.*, rectangular or hexagonal), and the pixel aspect ratio.

4.4 IPI derived non-image data types

The following data types are considered to be non-image data types:

— basic data types, as defined in clause 4.2;

— image annotation data types (*e.g.*, textual descriptions, graphical descriptions of boundaries, audio tracks of motion pictures);

— image-related data types (*e.g.*, colour look-up tables, histograms).

Non-image data types, unlike image attributes, are not necessarily bound to an image. Image annotation and image-related data types are described in the following paragraphs.

4.4.1 IPI derived image annotation data types

Broad categories of image annotation data types are intended to provide a flexible interface to other non-imaging standards. These data types can be integrated into image data types to allow for textual, auditory, and graphical descriptions of the image content according to some application-specific requirements. These IPI-derived image annotation data types must be constructed by an application program from the elementary data types discussed above.

Text, graphics, and audio data types are not defined within ISO/IEC 12087. These data types are referenced only via unique names. Their detailed structure must be defined by an application program in accordance to some other standards (*e.g.*, [ISO/IEC 8613], [3], [5]) and their consistent use has to be guaranteed by the application program.

4.4.2 IPI derived image-related non-image data types

These non-image data types are also constructed from the elementary data types described in clause 4.2.1. As with image attributes, these types are described in a textual way by outlining their required functionality; details of the specific delineations are given in clause 5 and clause 6.

feature list — a collection of pairs of elements. The first element of the pair is a coordinate of a pixel, while the second refers to a basic data type storing the associated feature. The feature list forms the basis of the value bounds collection described in clause 5 and clause 6.

histogram — a construction that counts the frequency with which pixel values occur with respect to some partition $\{V_1, \ldots, V_N\}$ of the total range of pixel values. The partition elements V_i $(i = 1, \ldots, N)$

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 0607759 208

could be defined as, for example, arrays of the lower and upper bounds of intervals along the axes of a multi-band image, or as sets containing the pixel values falling into the associated partition elements.

Familiar examples of histograms are counts of the number of pixels with respect to some partition amplitude ranges, *e.g.* 8-bit grey values. However, histograms, as defined here, could also measure higher-dimensional co-occurrence probabilities over multi-band images for cluster analysis or statistical modelling (Markov models) of multi-sensor input; or, in non-destructive testing applications, the number of regions in an image that match templates, thereby modelling structural defects.

Example operations on histograms are: check equality of two histograms, add an element to a histogram, get the current bounds of a partition element, set element counts to zero.

look-up table (LUT) — a structure that contains pairs of entries: the first entry of the pair is an input value to be matched, while the second contains a corresponding output value. Note that look-up tables are not restricted to mapping scalar onto scalar.

Look-up tables may be used for, for example, colour look-up tables for mapping scalar and non-scalar pixel data onto some colour representation, *e.g.* for display purposes using some colour coding scheme; coding and decoding schemes for packing non-scalar pixel values into some scalar representation and *vice versa*; and transfer functions correcting raw sensor data for some sensor characteristics or mapping raw sensor data onto some application-specific internal representation.

Example operations on look-up tables are: check the equality of two tables; check whether entry (x, y) is a member of the table; add an entry (x, y) to the table, if it is not already a member of the table; remove an entry (x, y) from the table, if it is a member; for input value x, if entry (x, y) is a member of the table, get output value y; get current table length.

neighbourhood — a mechanism for defining a region of an image; this region is not referenced to a particular position within an image. For example, a neighbourhood could be used for specifying an impulse response function for a convolution operator. Those particular representations of a neighbourhood that are supported by ISO/IEC 12087 are described in clause 5 and clause 6.

region of interest (ROI) — a Region of Interest (ROI) is defined as a general mechanism for pixel-by-pixel selection, yielding a Boolean value for each pixel location of the image. In principle, the representation of a ROI could be, for example, via geometrical models (for circles, cylinders, *etc.*), via coordinates (for rectangular regions), or via a Boolean array of the same dimensions as the image. Those particular representations of a ROI that are supported by ISO/IEC 12087 are described in clause 5 and clause 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 0607760 T2T

5 IPI-PIKS architecture

Clause 4 described a generic architecture for the representation and processing of images under which a number of imaging APIs and transport facilities of varying degrees of functionality and complexity can be derived. This clause presents the architecture of a specific delineation of the generic architecture which is used to support the imaging API known as the Programmer's Imaging Kernel System (IPI-PIKS).

5.1 IPI-PIKS imaging model

The imaging model used by IPI-PIKS is delineated from the generic model described in clause 4. However, the IPI-PIKS architectural specification provides specific delineations of the system operator and control functionality; in particular, the IPI-PIKS operator model explicitly defines the means of neighbourhood control and of image control, as indicated in figure 5.

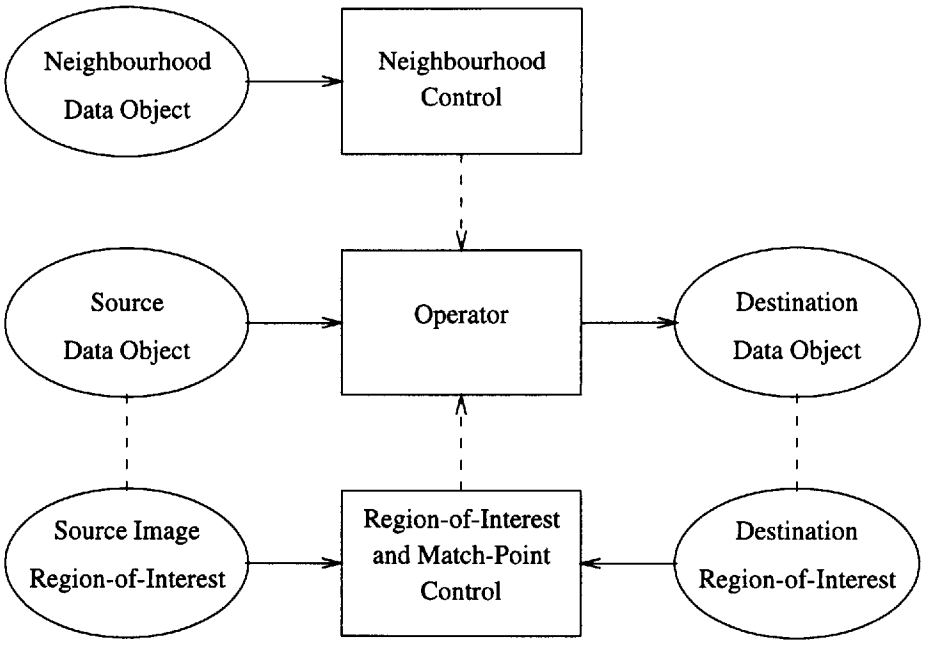


Figure 5 — The operator model used by IPI-PIKS

5.1.1 IPI-PIKS neighbourhood control

In IPI-PIKS, neighbourhood control is accomplished by means of a neighbourhood object, which consists of a fixed array of data values of identical data type. Such a neighbourhood array is typically much smaller in dimensional extent than an image. Common instances of an image neighbourhood object are an impulse response array and a morphological structuring element array. The nature of the neighbourhood control is operator-dependent. Further details of this topic are provided in ISO/IEC 12087-2.

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 0607761 966

5.1.2 IPI-PIKS image control

Image control is accomplished in IPI-PIKS by two means: ROI control of the source and destination images and match-point control.

An IPI-PIKS ROI may be conceptually thought of as an array of Boolean data type, each element of which is associated with a pixel of an image. Separate ROIs are associated with source and destination images. Jointly, source and destination ROIs determine which pixels of source images are to be processed and which pixels are recorded in destination images. Furthermore, a ROI could be used to determine how the pixels of an image are to be processed (*e.g.*, taking the logarithm of each pixel value within a ROI and the square root of every pixel value outside the ROI), though such a scheme is not yet used by any of the IPI-PIKS operators.

Each IPI-PIKS image has an associated match-point. Prior to processing, all source and destination images undergo translation to align their match-points. A ROI bound to an image undergoes the same translation as the image to which it is bound.

Further specification of IPI-PIKS image control is provided in ISO/IEC 12087-2.

5.2 IPI-PIKS system control

IPI-PIKS system control performs four basic functions:

— data object management

— operational synchronization

— element chaining

— error management

These functions are described in greater detail in the remainder of this clause.

5.2.1 Data object management

IPI-PIKS treats all image and image-related, non-image data as abstract data objects internally. In IPI-PIKS, the physical representation and storage methods of data objects are provided transparently to the application by an API implementation. The application need not be explicitly concerned with implementation methods such as

— numeric data representation (*e.g.*, integer or floating-point);

— computational precision (*e.g.*, 16- or 32-bit integers);

— data storage format (*e.g.*, pixel interleaved or tiled bands).

An IPI-PIKS implementation provides enquiry facilities concerning specific implementation methods but does not specify physical representations.

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

4851903 0607762 8T2

5.2.2 Operational synchronicity

IPI-PIKS provides for operation either synchronously or asynchronously. In the synchronous mode, each step of a sequence of processes must be completed before the next step is initiated. In the asynchronous mode, concurrent processing is permitted — provided, of course, that the input data are available to a process before that process is initiated.

5.2.3 Element chaining

In image processing, it is common to link a series of processes together to achieve some desired result. IPI-PIKS provides such a mechanism, called *element chaining*, whereby chains of functions may be explicitly identified and called for execution as higher-level entities. IPI-PIKS also provides a mechanism for naming chains and dynamically specifying their arguments.

5.2.4 Error management

In any computing environment, errors may occur. It is important that such errors are identified readily and handled gracefully. To this end, IPI-PIKS and IPI-IIF provide an error management mechanism which permits accurate error reporting but which does not usually abort processing.

5.3 IPI-PIKS basic data types

This clause defines specific representations of the generic data types described in clause 4 that are applicable to IPI-PIKS. Unless explicitly stated in this clause, the generic data types are *not* carried forward into IPI-PIKS.

Basic data types have been described generically in clause 4.2. IPI-PIKS supports only a subset of these generic data types, as defined in this clause.

5.3.1 IPI-PIKS elementary data types

IPI-PIKS supports the following elementary data types, as discussed in clause 4.2.1:

bit
Boolean
character
complex
enumerated
integer
null
real
state

All these types are consistent with annex 8.4.3. Two instances of the integer class are supported by IPI-PIKS:

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

4851903 0607763 739

signed integer — as defined in annex 8.4.3;

non-negative integer — as defined in annex 8.4.3 but with the additional restriction that only zero and positive integers may be represented.

In IPI-PIKS, the enumerated data type is used to specify the values of options in IPI-PIKS functional elements and data object attributes. Such options are limited in number — *e.g.*, 'up' or 'down.' Option lists that can potentially be extended are specified using a signed integer type — *e.g.*, adding an additional arithmetic function to the `dyadic` operator (see ISO/IEC 12087-2).

The IPI-PIKS complex data type is composed of a doublet of real data types, which represent the real and imaginary parts of a complex number.

The state data type is used, for example, to describe whether the use of match-points is enabled or disabled.

The description of data types given in this clause is not prescriptive: an implementation may use different internal representations for the data types provided these meet the conformance criteria discussed in clause 8 and elsewhere in ISO/IEC 12087. An implementation may also provide a data type at more than one precision, in which case they must be treated as separate types.

5.3.2 IPI-PIKS compound data types

IPI-PIKS supports the following compound data types, which are derived from the array, table, list, pointer, and record generic compound data types defined in clause 4.2.2. These tables indicate content only; ordering of elements within each table is not implied.

character string — as described in annex 8.4.3. The character string shall not be exchanged by IPI-IIF.

choice — as described in annex 8.4.3. The choice shall not be exchanged by IPI-IIF.

ID_ARRAY — an array of data object identifiers that is used in the creation and manipulation of compound images in IPI-PIKS. An ID_ARRAY object is denoted by ID_ARRAY and consists of the following fields:

ENTRY NAME	DESCRIPTION
length, *N*	number of entries in array
identifier array	array of identifiers

Objects of type ID_ARRAY are internal to IPI-PIKS and may not be exchanged by IPI-IIF.

ID_LIST — a list of data object identifiers that is used in the creation and manipulation of compound images in IPI-PIKS. An ID_LIST object is denoted by ID_LIST and consists of the following fields:

ENTRY NAME	DESCRIPTION
first item	identifier of first list_item

where list_item consists of the following fields:

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

4851903 0607764 675

forward link
prior link
identifier

Objects of type ID_LIST are internal to IPI-PIKS and may not be exchanged by IPI-IIF.

ID_RECORD — a record of data object identifiers that is used in the creation and manipulation of compound images in IPI-PIKS. An ID_RECORD object is denoted by ID_RECORD and consists of the following fields:

ENTRY NAME	DESCRIPTION
length, N	value
name tuple	tuple of record names
identifier tuple	tuple of identifiers

where name tuple consists of the following fields:

N
'TC'
record name
⋮

and identifier tuple consists of the following fields:

N
'ID'
identifier
⋮

Objects of type ID_RECORD are internal to IPI-PIKS and may not be exchanged by IPI-IIF.

pixel record — an aggregate of B pixels, with each pixel value being of one of the elementary data types supported by IPI-PIKS. A pixel record consists of the following fields:

ENTRY NAME	DESCRIPTION
length, N	value
data type	pixel type B-tuple
value 1	value
value 2	value
⋮	⋮
value B	value

where the data type B-tuple consists of the following fields:

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

4851903 0607765 501

B
'TC'
CHOOSE{'BC','NC','SC','RC','CC'}
CHOOSE{'BC','NC','SC','RC','CC'}
⋮
CHOOSE{'BC','NC','SC','RC','CC'}

pointer — as described in annex 8.4.3. The pointer shall not be exchanged by IPI-IIF.

tuple — an aggregate of object attributes combined with a collection of values of the same elementary data type, which represents IPI-PIKS functional element parameters and image object parameters.

A tuple object consists of the following fields:

ENTRY NAME	DESCRIPTION
tuple data size	number of values
data type code	CHOOSE{'BC','NC','SC','RC','CC','TC'}
value 1	value
value 2	value
⋮	⋮
value *P*	value

This definition does not provide a constraint on the implementation of the tuple, since it is accessed only via IPI-PIKS tools.

The definitions of the data type codes are given in table 1. The other compound data types defined in clause 4 are not available within IPI-PIKS in their general form. However, certain IPI-PIKS data types, both image and non-image, are derived from these generic compound data types.

CODE STRING	MEANING
'BC'	Boolean code
'EC'	enumerated code
'NC'	non-negative integer code
'SC'	signed integer code
'RC'	real number code
'CC'	complex number code
'TC'	text (character) string code

Table 1 — Codes for the externally-visible representations of IPI-PIKS-specific data types

NOTES

6 In some language bindings, it may not be possible to represent the tuple type directly.

7 In IPI-PIKS, the choice compound data type is used implicitly, via the CHOOSE{·} operator, as a means of choosing a subset of the generic data types for the IPI-PIKS-specific data types.

8 In IPI-PIKS, the pointer compound data type is used to abstractly identify a data object, a variable of a particular data 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 0607766 448

An *identifier* uniquely denotes an IPI-PIKS data object.

NOTE 9 Identifiers may be implemented as pointers.

5.4 IPI-PIKS derived image data descriptions

This clause contains a number of tables that describe the content of certain data aggregates. These tables indicate content only; ordering of elements within each table is not implied.

5.4.1 IPI-PIKS derived data types

IPI-PIKS supports several image data types that are derived from the compound data types defined generically in clause 4. An IPI-PIKS image has the following attributes associated with it:

ATTRIBUTE	DESCRIPTION
representation	representation description identifier
channel	channel description identifier
colour	colour description identifier
control	control description identifier

The details of these image-related descriptions are defined in clause 5.4.4.

5.4.2 IPI-PIKS compound image data types

An *image* is considered by IPI-PIKS to be a five-dimensional (5D) data structure, where the indices have the meanings indicated in table 2. The designators X, Y, Z, T and B in this table represent the maximum sizes along each dimension. However, an image is not necessarily an array (see clause 4.2.2) because the data type may vary with the b index. Within each band (*i.e.*, for any combination of x, y, z, t values but a fixed b value), the data type must be homogeneous. This heterogeneity across the band dimension is intended to facilitate the combination of images from several, different sensors into a single IPI-PIKS image. Furthermore, to allow meaningful names to be associated with particular bands (*e.g.*, `'red'`), IPI-PIKS provides access mechanisms that accept a band specification that is either numeric or textual.

INDEX RANGE	INDEX MEANING
$0 \leq x < X$	spatial columns (horizontal)
$0 \leq y < Y$	spatial rows (vertical)
$0 \leq z < Z$	depth pixel planes
$0 \leq t < T$	temporal pixel planes
$0 \leq b < B$	colour bands, spectral bands, or generic pixel planes

Table 2 — Dimensions of an IPI-PIKS Data Object

The array sizes X, Y, Z and T are constant and do not vary with the b index.

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

4851903 0607767 384

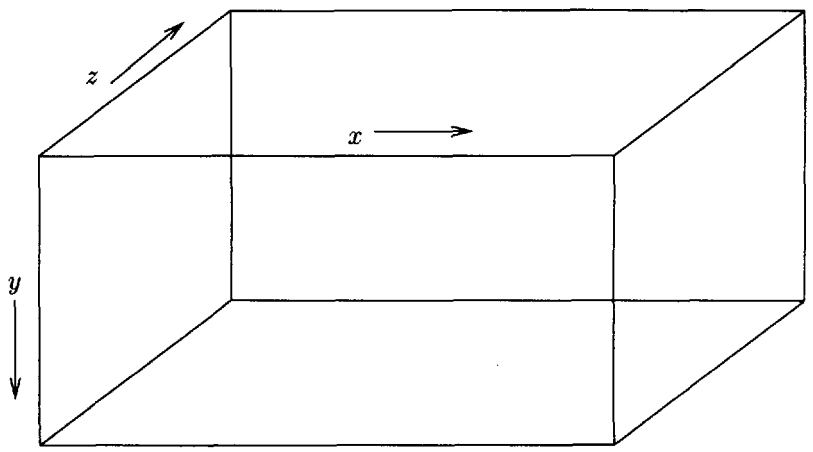


Figure 6 — Relationship Between a Physical Volume and IPI-PIKS Horizontal, Vertical, and Depth Coordinates

NOTE 10 Some of the image dimensions may be unpopulated, *e.g.* for a monochrome image $Z = T = B = 1$.

A two-dimensional pixel array involving only two image indices is called a pixel plane. A pixel plane involving only the (x, y) indices is termed an x-y pixel plane. Conceptually, a five-dimensional image array can be considered as being organized as an aggregate of depth, time, and band x-y pixel planes.

Wherever applicable, IPI-PIKS specifies relationships between physical imaging coordinates (volume, time, wavelength) and the semantic indices (x, y, z, t, b) of an IPI-PIKS image. For example, figure 6 illustrates the physical relationship between a physical volume (horizontal, vertical, depth) and the IPI-PIKS volume indices (x, y, z). The volumetric origin is at the upper left front corner of the cuboid. Index x increases positively left-to-right, horizontally. Index y increases positively top-to-bottom, vertically. Index z increases positively front-to-rear. The temporal index t increases positively over a past-to-future time base. The index b, when used to designate spectral image bands numerically, increases positively from a lower to a higher wavelength.

NOTES

11 The x left-to-right and y top-to-bottom index progression adopted by IPI-PIKS is consistent with the presentation format of most imaging systems.

12 In many computer graphics software systems, the y index progression is bottom-to-top.

13 IPI-PIKS provides for the specification of a generic five-dimensional image in which the (x, y, z, t, b) have no semantic meaning.

14 To avoid confusion, the term 'band' is reserved for the index name of colour and multispectral images only.

5.4.3 Composite images

The 5D representation of images employed by IPI-PIKS is adequate for the majority of applications; however, when a task naturally lends itself to highly structured image representations, the 5D representation may not be appropriate. For this reason, a facility is provided by which several 5D images may be combined into

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

4851903 0607768 210

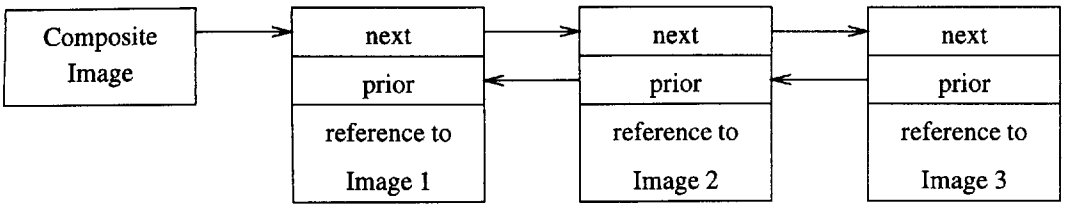


Figure 7 — Aggregation of Image References into a List

a *composite image*. This aggregation mechanism does not increase the dimensionality of IPI-PIKS images above five, but does permit IPI-PIKS images to be loosely associated. Aggregation aims to minimize memory requirements, to avoid the duplication of image data, and (if necessary) to permit the same image data to be simultaneously combined into different logical structures.

The location and representation of an image within IPI-PIKS is necessarily not visible to the application; the application program merely has access to a reference to an image. The image aggregation mechanism provides the application program with a means for combining a number of image references into a single entity. This entity might be any of the array, list, or record data types described in clause 4.2.2 applied to image identifiers.

Functions are provided within IPI-PIKS to perform aggregation, de-aggregation, and access to the individual images of an aggregation. These are described in ISO/IEC 12087-2. Note that the image aggregation facilities are only required to be available in a full implementation of IPI-PIKS.

The IPI-PIKS operators do not themselves process composite images; the application program must explicitly extract image references from aggregate data structures and pass them to operators. However, when composite images are used in conjunction with operator chaining, the program is able to effectively define a loop over a set of images in a portable manner.

To provide an example of how these image aggregation facilities may be used, figure 7 illustrates three image references that have been aggregated into a (doubly-linked) list of images. The first of the three images is referred to by an "aggregation reference," and each member of the list indicates where the next and prior list members are located.

5.4.4 IPI-PIKS image object attributes

Only a subset of the image attribute descriptors described in clause 4 are supported within IPI-PIKS; those attributes that are supported are also constrained in their content. The supported attribute descriptors are specified below. Further details concerning these attributes and the functionality associated with them are given in ISO/IEC 12087-2.

representation attribute — this attribute contains information concerning the image dimensions, and the representation of the image bands, where the image structure codes are defined in annex 8.4.3.

ENTRY NAME	DESCRIPTION
image size	image size 5-tuple
band data type	pixel data type B-tuple
image structure	image structure code

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

4851903 0607769 157

where the image structure codes are defined in annex 8.4.3. The image size 5-tuple consists of the following fields:

5
'NC'
X size value
Y size value
Z size value
T size value
B size value

and the pixel data type B-tuple consists of the following fields:

B
'TC'
CHOOSE{'BC','NC','SC','RC','CC'}
CHOOSE{'BC','NC','SC','RC','CC'}
⋮
CHOOSE{'BC','NC','SC','RC','CC'}

where CHOOSE{·} represents the choice data generator, which is introduced in clause 5.3.2. The permitted values of the image structure code are defined in annex 8.4.3.

channel attribute — this attribute gives details of the pixel precision of each channel of an IPI-PIKS image. There are B entries in this attribute in a B-band image.

ENTRY NAME	DESCRIPTION
band pixel precision	bits/pixel B-tuple

where the channel attribute B-tuple consists of the following fields:

B
'NC'
value for band 1
value for band 2
⋮
value for band B

colour attribute — this attribute describes, for colour images only, the colour representation used. The permissible colour representations are defined in annex 8.4.3.

ENTRY NAME	DESCRIPTION
colour representation	colour representation code
white point	specification of white point or NULL

control attribute — this attribute carries references to the various image-related data structures that determine how the image will be affected by processing 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

4851903 0607770 979

ENTRY NAME	DESCRIPTION
ROI identifier	ROI identifier
ROI offset	ROI offset 5-tuple
match point	match point 5-tuple

The ROI offset 5-tuple consists of the following fields:

5
'SC'
x index ROI offset value
y index ROI offset value
z index ROI offset value
t index ROI offset value
b index ROI offset value

The match point 5-tuple consists of the following fields:

5
'RC'
x index match point value
y index match point value
z index match point value
t index match point value
b index match point value

An implementation of an IPI-PIKS binding is not required to represent each of the image attributes in a separate data structure; their physical representation is entirely the prerogative of the implementation, within the constraints outlined above. It is only necessary that an application program be able to enquire the values of the attributes.

5.5 IPI-PIKS derived non-image data structures

The IPI-PIKS non-image data objects are a derived subset, in terms of functionality, of the generic data objects described in clause 4. There are two classes of non-image data objects:

— data objects that are generated by an IPI-PIKS element and used by some other IPI-PIKS element as input data;

— data objects that are capable of being imported or exported between IPI-PIKS and the application or IPI-IIF.

NOTE 15 Some IPI-PIKS elements generate data objects that are not used by another IPI-PIKS element or exported to IPI-IIF. Such data objects are only returned to an application. Their data structure is defined in the description of a particular 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 0607771 805

IPI-PIKS supports the following non-image data objects, which are derived from the pointer, array, table, list, and record generic compound data types defined in clause 4.2.2.

chain — a data type which is used to store the identifier of a chain in IPI-PIKS. Objects of type chain are internal to IPI-PIKS and may not be exchanged by IPI-IIF.

histogram — an aggregate of object attributes combined with an array of E entries (E being a positive integer), where the content of each entry is a non-negative integer describing the number of pixels, among some collection of pixels, whose amplitudes lie in one of E bins, which are uniformly quantized between lower and upper amplitude bounds. The array has a specified size. A histogram object is denoted by `HIST` and consists of the following fields:

ENTRY NAME	DESCRIPTION
array size, E	value
lower amplitude bound	value
upper amplitude bound	value
histogram	data

look-up table — an aggregate of object attributes combined with a table of the same elementary data types. The table has a specified size. A lookup table object is denoted by `LUT` and consists of the following fields:

ENTRY NAME	DESCRIPTION
table entries	number of entries in table
table bands	number of output values for each table entry
input data type	`CHOOSE{'NC','SC'}`
output data type	`CHOOSE{'BC','NC','SC','RC','CC'}`
table	table data

matrix — a two-dimensional array of values of integer, real or complex data type with associated descriptive information. A matrix object is denoted by `MATRIX` and consists of the following fields:

ENTRY NAME	DESCRIPTION
column size	value
row size	value
matrix data type	`CHOOSE{'NC','SC','RC','CC'}`
array	array data

neighbourhood array — an aggregate of object attributes with an array of up to five dimensions of values of the same elementary data type, which represents a neighbourhood processing structure. A neighbourhood array object, denoted by `'NBHOOD_ARRAY'`, consists of the following fields:

ENTRY NAME	DESCRIPTION
data type	`CHOOSE{'BC','NC','SC','RC','CC'}`
neighbourhood size	neighbourhood size 5-tuple
key pixel	key pixel 5-tuple
scale factor	value
semantic label	one of: generic, dither, impulse response, mask, structuring element `CHOOSE{'GL','DL','IL','ML','SL'}`
neighbourhood	neighbourhood array data

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

4851903 0607772 741

where the key pixel and scale factor are explained in annex G of ISO/IEC 12087-2.

The neighbourhood size 5-tuple consists of the following fields:

5
'NC'
first index array size value
second index array size value
third index array size value
fourth index array size value
fifth index array size value

The key pixel 5-tuple consists of the following fields:

5
'NC'
first index key pixel value
second index key pixel value
third index key pixel value
fourth index key pixel value
fifth index key pixel value

NOTES

16 The key pixel offset is measured with respect to the origin of the neighbourhood array.

17 Unspecified values in the tuple are represented as NULLs.

ROI data objects — Further details on the semantic use of all ROI data objects are given in ISO/IEC 12087-2.

ROI array — an aggregate of object attributes with a five dimensional array of values of the Boolean data type, which represents a region-of-interest mask. A ROI array object is denoted by `ROI_ARRAY` and consists of the following fields:

ENTRY NAME	DESCRIPTION
ROI size	ROI size 5-tuple
ROI	ROI array

The ROI size 5-tuple consists of the following fields:

5
'NC'
X size value
Y size value
Z size value
T size value
B size value

and is the size of a box which bounds the ROI.

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

4851903 0607773 688

NOTE 18 The specification of a ROI data object as a 5D array does not impose any requirement upon an implementation to store the ROI array object as an array.

ROI coordinate collection — a collection of object attributes combined with a list of 5D coordinates of the non-negative integer data type, which specifies a ROI mask. A ROI coordinate collection object is denoted by ROI_COORD and consists of the following fields:

ENTRY NAME	DESCRIPTION
ROI size	ROI size 5-tuple
collection size	value
polarity	CHOOSE{'TRUE', 'FALSE'}
ROI	collection of coordinates

The ROI size 5-tuple consists of the following fields:

5
'NC'
X size value
Y size value
Z size value
T size value
B size value

and is the size of a box which bounds the ROI.

A ROI coordinate collection of E entries consists of the following fields:

$$\begin{array}{ccccc} x(0) & y(0) & z(0) & t(0) & b(0) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x(e) & y(e) & z(e) & t(e) & b(e) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x(E-1) & y(E-1) & z(E-1) & t(E-1) & b(E-1) \end{array}$$

ISO/IEC 12087-2 contains a description of the construction process for a ROI coordinate list object.

ROI elliptical — an aggregate of object attributes combined with a sequence of values, which specifies a region-of-interest mask in terms of a filled elliptical shape. A ROI elliptical object is denoted by ROI_ELLIP and consists of the following fields:

ENTRY NAME	DESCRIPTION
ROI size	ROI size 5-tuple
ellipse centre	ellipse centre 5-tuple
ellipse axis length	axis length 5-tuple
index manipulate	index manipulation 5-tuple
dimension	CHOOSE{'1D','2D','3D','4D','5D'}
polarity	CHOOSE{'TRUE', 'FALSE'}

The ROI size 5-tuple consists of the following fields:

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

4851903 0607774 514

5
'NC'
X size value
Y size value
Z size value
T size value
B size value

and is the size of a box which bounds the ROI.

The ellipse centre 5-tuple consists of the following fields:

5
'RC'
x index ellipse centre
y index ellipse centre
z index ellipse centre
t index ellipse centre
b index ellipse centre

The axis length 5-tuple consists of the following fields:

5
'RC'
x index ellipse axis length
y index ellipse axis length
z index ellipse axis length
t index ellipse axis length
b index ellipse axis length

The index manipulate 5-tuple consists of the following fields:

5
'NC'
X index manipulate value
Y index manipulate value
Z index manipulate value
T index manipulate value
B index manipulate value

ISO/IEC 12087-2 contains a description of the construction process for a ROI elliptical object and its properties.

ROI generic — an identifier of one of the specific ROI data objects. A ROI generic object is denoted by ROI_GEN; it consists of the following field:

ENTRY NAME	DESCRIPTION
ROI	ROI identifier
ROI size	ROI size 5-tuple

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

4851903 0607775 450

ROI polygon sequence — an aggregate of object attributes combined with a sequence of two-dimensional coordinates of the non-negative integer data type which specifies a region-of-interest mask in terms of a filled polygon. A ROI polygon list object is denoted by ROI_POLY consists of the following fields:

ENTRY NAME	DESCRIPTION
ROI size	ROI size 5-tuple
sequence size	number of 2D coordinates in sequence
index manipulate	index manipulation 5-tuple
ROI	polygon sequence
polarity	CHOOSE{'TRUE', 'FALSE'}

The ROI size 5-tuple consists of the following fields:

5
'NC'
X size value
Y size value
Z size value
T size value
B size value

and is the size of a box which bounds the ROI.

A ROI polygon sequence of E entries consists of the following fields:

$$\begin{matrix} x(0) & y(0) \\ \vdots & \vdots \\ x(e) & y(e) \\ \vdots & \vdots \\ x(E-1) & y(E-1) \end{matrix}$$

The index manipulate 5-tuple consists of the following fields:

5
'NC'
X index manipulate value
Y index manipulate value
Z index manipulate value
T index manipulate value
B index manipulate value

ISO/IEC 12087-2 contains a description of the construction process for a ROI polygon sequence object and its properties.

ROI rectangular — an aggregate of object attributes combined with a sequence of values, which specifies a region-of-interest mask in terms of a filled rectangular shape. A ROI rectangular object is denoted by ROI_RECT; it consists of the following fields:

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

4851903 0607776 397

ENTRY NAME	DESCRIPTION
ROI size	ROI size 5-tuple
start position	start position 5-tuple
end position	end position 5-tuple
index manipulate	index manipulation 5-tuple
dimension	CHOOSE{'1D','2D','3D','4D','5D'}
polarity	CHOOSE{'TRUE', 'FALSE'}

The ROI size 5-tuple consists of the following fields:

5
'NC'
X size value
Y size value
Z size value
T size value
B size value

and is the size of a box which bounds the ROI.

The start position 5-tuple consists of the following fields:

5
'NC'
x index start position
y index start position
z index start position
t index start position
b index start position

The end position 5-tuple consists of the following fields:

5
'NC'
x index end position
y index end position
z index end position
t index end position
b index end position

The index manipulate 5-tuple consists of the following fields:

5
'NC'
X index manipulate value
Y index manipulate value
Z index manipulate value
T index manipulate value
B index manipulate 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 12087-2 contains a description of the construction process for a ROI rectangular object and its properties.

static array — an aggregate of object attributes with an array of up to five dimensions of values of the same elementary data type. A static array object, denoted by STATIC_ARRAY, consists of the following fields:

ENTRY NAME	DESCRIPTION
array size	array size 5-tuple
semantic label	one of: generic, power spectrum, transfer function, windowing function CHOOSE{'GL', 'PL', 'TL', 'WL'}
data type	CHOOSE{'BC','NC','SC','RC','CC'}
array	array data

The array size 5-tuple consists of the following fields:

5
'NC'
X size value
Y size value
Z size value
T size value
B size value

value bounds collection — a data structure that is used only by the IPI-PIKS routine of the same name. It consists of an aggregate of object attributes combined with an ordered list of entries $[x, y, z, t, b, S(x, y, z, t, b)]$, which contains the image index set $[x, y, z, t, b]$ of non-negative data types ('NC') and the associated pixel value $S(x, y, z, t, b)$ for each pixel whose value lies within some specified amplitude bounds. A value bounds collection is denoted by VALUE_BOUNDS and consists of the following fields:

ENTRY NAME	DESCRIPTION
collection size	value
lower amplitude bound	value
upper amplitude bound	value
pixel data type	CHOOSE{'BC','NC','SC','RC'}
collection	value bounds collection data

A value bounds list of E entries consists of the following fields:

$$\begin{array}{ccccc|c} x(0) & y(0) & z(0) & t(0) & b(0) & S[x(0), y(0), z(0), t(0), b(0)] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x(e) & y(e) & z(e) & t(e) & b(e) & S[x(e), y(e), z(e), t(e), b(e)] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x(E-1) & y(E-1) & z(E-1) & t(E-1) & b(E-1) & S[x(E-1), y(E-1), z(E-1), t(E-1), b(E-1)] \end{array}$$

virtual register — a data type which supports storage of values within IPI-PIKS and can be used for communication, synchronization, and process control within IPI-PIKS only, *i.e.* cannot be interchanged by IPI-IIF. It consists of the following fields:

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

4851903 0607778 16T

ENTRY NAME	DESCRIPTION
data type code	CHOOSE{'BC', 'NC', 'SC', 'RC', 'CC', 'TC', 'ID', 'IP'}
status flag	VREG_SET or VREG_CLEARED
value	value

An implementation of an IPI-PIKS binding is not required to represent each of the image attribute descriptors in a separate data structure; their physical representation is entirely the prerogative of the implementation, within the constraints outlined above. It is only necessary that an application program be able to enquire the values of the attribute descriptors.

5.6 IPI-PIKS data pragmata

The IPI-PIKS data types defined in this clause are abstractions conforming to some special mathematical rules. Each physical implementation of these IPI-PIKS data types has its own internal data types. An implementation of IPI-PIKS data types is a particular mapping from IPI-PIKS data types to internal data types.

A data pragmata is a description of a mapping between an abstract data type and a physical data type. IPI-PIKS does not specify data pragmata, but ISO/IEC 12087 does provide guidelines for data type mappings, as discussed in the following paragraph.

IPI-PIKS implementations may use any internal representation and precision for the 'BC', 'NC', 'SC', 'RC', and 'CC' elementary data types provided that their IPI-PIKS implementation meets minimal levels of conformance (see clause 8). In typical implementations, the 'NC' data type will be represented by an unsigned or positive integer. The signed integer data type is common in many implementations. Some implementations may choose to utilize signed integers in two's complement format for the real arithmetic data type, while others may use a floating point representation. It should be noted that an implementation may choose a mixed representation in which some images are signed integer, and others are floating point, for example. It is also possible within the IPI-PIKS abstract data type concept to support images of the same elementary data type with different levels of precision.

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

4851903 0607779 0T6

6 IPI-IIF-specific architecture

6.1 IPI-IIF imaging model

The imaging model used by IPI-IIF is identical to that described in clause 5, which is delineated from the generic model of clause 4. IPI-IIF fulfills two purposes: the first is the definition of the IPI-IIF data format for interchange of digital images; the data format is specified syntactically by means of ASN.1 (see ISO/IEC 12087-3). Secondly, IPI-IIF incorporates a gateway whose purpose is to control the import and export of image data to and from application programs and to and from IPI-PIKS (see figure 3). This is achieved via an IPI-IIF "gateway," the functionality of which is classified into the following categories:

import/export to application — these functions allow for the import and export of image data to and from the IPI-IIF gateway by the application program.

compose/decompose — these functions provide facilities for inter-converting between the restricted data types acceptable to IPI-PIKS and the more structured data types described in clause 4 and in this clause. The 'compose' function synthesizes more structured data types from simpler IPI-PIKS-compatible data types, while the 'decompose' operation extracts the simpler IPI-PIKS data types from the more structured ones. Hence, these functions effectively form a conversion mechanism between structured and unstructured representations.

parse/generate — these functions facilitate the reading and writing of the IPI-IIF data stream according to the IPI-IIF data format and IPI-IIF syntax described in ISO/IEC 12087-3. The 'parse' function analyses the IPI-IIF data stream and passes on the identified IPI-IIF data tokens to the internal data structures of the IPI-IIF gateway. The 'generate' function synthesizes an IPI-IIF data stream according to the IPI-IIF syntax from the image representation used within the IPI-IIF gateway. (This IPI-IIF gateway representation need not be the same as that used by IPI-PIKS.)

image structure manipulation — these functions provide the ability to attach attributes to images, *etc.*

gateway control functions — these functions provide explicit control over the action of an IPI-IIF gateway (*e.g.*, opening and closing).

These categories of functionality are not necessarily orthogonal: for example, functions in the import/export class may invoke functions in the compose/decompose category, which may themselves utilize parse/generate functions.

To perform these functions, IPI-IIF has an API which may be used by an application program. These API functions are defined in Part 3 of ISO/IEC 12087.

There is a defined relationship between IPI-IIF and IPI-PIKS, which is described in clause 7.

6.2 IPI-IIF basic data types

All the fundamental data types defined in clause 4.2 may be represented and interchanged by IPI-IIF, but only when built into a derived data 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 0607780 818

6.3 IPI-IIF derived data types

IPI-IIF is capable of transferring all the data types used by IPI-PIKS unless otherwise stated in clause 5.

6.3.1 IPI-IIF derived image data types

IPI-IIF is capable of transferring both the image data types used by IPI-PIKS and the structured data types described in clause 4.3.2. Indeed, it allows the level of structuring to be arbitrarily complex. For further information, refer to ISO/IEC 12087-3.

6.3.2 IPI-IIF image attributes

IPI-IIF supports the following image attributes:

ATTRIBUTE	DESCRIPTION
representation	representation description identifier
channel	channel description identifier
colour	colour description identifier
control	control description identifier
free-form	free-form description identifier

which are described briefly in the following paragraphs. The following attribute tables indicate required information but do not constrain their placement within the IPI-IIF syntax. Full details of the attributes are given in ISO/IEC 12087-3.

channel attribute — this attribute consists of three fields:

ENTRY NAME	DESCRIPTION
channel type	one of: bi-level, halftone, grey-value, colour, transparency, feature, application-specific CHOOSE{'BL','HT',GV','CO','TP','FE','AS'}
band pixel precision	number of bits/pixel
scanning characteristics	one of: scan type, gamma correction, quantization depths, transfer function label, NULL CHOOSE{'ST','GC','QD','TL'}

For an IPI-PIKS image, the type is always grey-value.

colour attribute — this attribute consists of the following fields:

ENTRY NAME	DESCRIPTION
colour system	one of the colour systems described in annex 8.4.3
colour space	one of the colour spaces described in annex 8.4.3
white point	the specification of a white point in CIE-XYZ system of the 1931 standard (see annex 8.4.3) or NULL
test colours	the specification of a sequence of test colours associated with particular regions of the image or 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 0607781 754

All fields in the colour attribute are optional.

control attribute — this attribute contains the following fields:

ENTRY NAME	DESCRIPTION
ROI identifier	identifier of a ROI
ROI offset	specification of a ROI offset
match point	specification of a match point

free-form attribute — this attribute consists of an arbitrary stream of data. The content of this attribute is not interpreted by IPI-IIF.

representation attribute — this attribute contains the following fields:

ENTRY NAME	DESCRIPTION
image size	specification of the sizes of the image
band data type	specification of the data type of the bands of the image
image structure code	a character string containing one of the image structure codes defined in annex 8.4.3
coordinate system	a description of the coordinate system in which the image is represented or NULL
dimension mappings	a sequence of dimension mappings, each of which provides a physical description of the recording system for one dimension of the image or NULL
metric transformation	one of: a standard transformation (*e.g.*, quincunx), a transformation matrix, a list of points, or NULL

where a dimension mapping consists of the following information:

identifier
coordinate axis
domain (*e.g.*, spatial or spatial frequency)
unit of measurement
origin of measurement system
scale factor

6.3.3 IPI-IIF derived non-image data types

6.3.3.1 IPI-IIF image annotation data types

IPI-IIF provides facilities for the transfer of text, graphics, and audio annotation of images via the *ImageAnnotationDataTypes* entity of IPI-IIF. This takes the values *TextDataDescriptor, GraphicsDataDescriptor,* or *AudioDataDescriptor* for the three respective types of annotation. For further details, refer to ISO/IEC 12087-3.

6.3.3.2 IPI-IIF image-related non-image data types

IPI-IIF defines the following image-related non-image data types:

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

4851903 0607782 690

feature list — this consists of the following fields:

ENTRY NAME	DESCRIPTION
number of coordinates	number of entries in the feature list
coordinate and feature	a sequence of feature descriptions

where each feature description has the following form:

coordinate of the feature in the image
feature number

histogram — this consists of the following fields:

ENTRY NAME	DESCRIPTION
number of partition classes	number of entries in the histogram
partition class data type	data type of the pixels of the image from which the histogram is derived
partition classes	a sequence of partition classes

Each partition class consists of the following fields:

lower boundary of partition
upper boundary of partition
count

look-up table — this consists of the following fields:

ENTRY NAME	DESCRIPTION
number of entries	number of entries in the look-up table
number of bands	number of output values for each table entry
input data type	the data type that will be used to index the table
output data type	the data type that will be output from the table
table entries	a sequence of look-up table entries

where each table entry consists of the following fields:

input value
output value

matrix — this consists of the following fields:

ENTRY NAME	DESCRIPTION
column size	value
row size	value
matrix data type	CHOOSE{'NC','SC','RC','CC'}
array	array identifier

neighbourhood — this consists of the following fields:

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

ENTRY NAME	DESCRIPTION
neighbourhood size	specification of the neighbourhood size
key pixel	specification of the key pixel
scale factor	value used for re-scaling the neighbourhood
semantic label	one of: generic, dither, impulse response, mask, structuring element CHOOSE{'GL','DL','IL','ML','SL'}
data type	CHOOSE{'BC','NC','SC','RC','CC'}
neighbourhood	specification of the neighbourhood identifier

region of interest — this may be represented in a number of ways:

ROI array — this is a generalization of the IPI-PIKS ROI array.

ENTRY NAME	DESCRIPTION
number of dimensions	number of dimensions of the ROI
dimension descriptors	sequence of dimension descriptions
ROI size	size of the ROI
ROI	identifier of a Boolean array

ROI elliptical — this corresponds to the IPI-PIKS ROI elliptical.

ENTRY NAME	DESCRIPTION
ROI size	size of the elliptical ROI
ellipse centre	ellipse centre
ellipse axis length	axis length
index manipulate	index manipulation specification
dimension	dimensionality of the elliptical ROI
polarity	CHOOSE{'TRUE', 'FALSE'}

ROI interval — this is a generalization of the IPI-PIKS ROI rectangular.

ENTRY NAME	DESCRIPTION
number of dimensions	number of dimensions of the ROI
ROI size	size of the ROI
lower boundary	vector that gives the lower boundary of the ROI in each dimension
upper boundary	vector that gives the upper boundary of the ROI in each dimension
index manipulate	index manipulation specification
polarity	CHOOSE{'TRUE', 'FALSE'}

ROI polygon — this corresponds to the IPI-PIKS ROI polygon sequence.

ENTRY NAME	DESCRIPTION
ROI	identifier of collection of coordinates
ROI size	ROI size
index manipulate	index manipulation specification
collection size	number of coordinate values that define the polygon ROI
polarity	CHOOSE{'TRUE', 'FALSE'}

ROI set of coordinates — this is a generalization of the IPI-PIKS ROI coordinate collection.

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

4851903 0607784 463

ENTRY NAME	DESCRIPTION
number of dimensions	number of dimensions of the ROI
ROI size	size of the ROI
number of coordinates	number of coordinates in the ROI
coordinates	identifier of coordinate values
polarity	CHOOSE{'TRUE', 'FALSE'}

static array — an aggregate of object attributes with an array of up to five dimensions of values of the same elementary data type. A static array object, denoted by STATIC_ARRAY, consists of the following fields:

ENTRY NAME	DESCRIPTION
array size	size of array
semantic label	one of: generic, power spectrum, transfer function, windowing function CHOOSE{'GL', 'PL', 'TL', 'WL'}
data type	CHOOSE{'BC','NC','SC','RC','CC'}
array	array data

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

4851903 0607785 3TT

7 Relationship between IPI-PIKS and IPI-IIF

To ensure that IPI-PIKS and IPI-IIF correctly interact, it is necessary to specify the relationship between the two parts, thereby ensuring mutual, consistent data exchange and control. This clause describes this relationship. Implementations of IPI-PIKS and IPI-IIF must conform with this specification, in both generic and specific terms; such conformance is intended to permit, for example, IPI-PIKS and IPI-IIF implementations to be obtained from different suppliers.

The standardized interface between IPI-PIKS and IPI-IIF is a data stream that conforms to the ASN.1 specification [ISO/IEC 8824:1990] described in greater detail in ISO/IEC 12087-3. As figure 3 shows, the parsing (for import to IPI-PIKS) and generation (for export from IPI-PIKS) of the data stream lie within the domain of IPI-PIKS. This parsing and generation facility within IPI-PIKS is in addition to the parsing and generation abilities of IPI-IIF. However, the data that are passed across the IPI-PIKS/IPI-IIF interface must contain only IPI-PIKS data types; images and related information with a higher degree of structure in the IPI-IIF domain must be reduced to IPI-PIKS-compatible data types. These IPI-PIKS data types form a strict subset of the generic data types described in clause 4 and of the range of data types supported by IPI-IIF (see clause 6). The mapping between IPI-PIKS data types and the IPI-IIF data types in its IIF-PIKS profile is given in ISO/IEC 12087-3. When it is necessary to import an image over the IPI-PIKS–IPI-IIF ASN.1 data stream that may not be represented by an IPI-PIKS data type, the IPI-IIF import/export mechanism that resides within IPI-PIKS (figure 3) must perform the necessary conversion, with potential loss of information.

One consequence of the restriction on the data types that may be interchanged between IPI-PIKS and IPI-IIF is that application programs which need to import and export data types or objects that are not supported by IPI-PIKS must do so *outside* the import/export mechanisms defined by the API presented in ISO/IEC 12087-2. However, ISO/IEC 12087-3 defines an API for the interchange of these more structured types of information directly with the application program. Hence, it is possible for an application to represent, interchange, and process such extra-IPI-PIKS data in a program that conforms to ISO/IEC 12087.

ISO/IEC 12087 also permits an implementation-dependent data stream to exist between IPI-PIKS and IPI-IIF: the nature of such a data stream is not defined within ISO/IEC 12087. This feature is intended to allow implementations that include both IPI-PIKS and IPI-IIF to use a more efficient communication mechanism than an ASN.1 data stream. However, dependence on a non-standardized data stream may result in non-portable applications.

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

4851903 0607786 236

8 Conformance

This Standard specifies the functional capability of an IPI implementation. Where an implementation claims to support a function defined in this Standard, the behaviour of the implementation shall match the function definition.

8.1 Conformance of functionality

It is permissible for a Standard-conforming IPI implementation to implement all the necessary non-IPI-IIF functionality of IPI-PIKS without any of the functionality of IPI-IIF. Similarly, a Standard-conforming IPI implementation may implement all the functionality of IPI-IIF without any of the functionality of IPI-PIKS. Requirements for IPI-PIKS and IPI-IIF functionality are determined entirely by the profile mechanism (see clause 8.4).

An implementation conforms to the IPI standard if all of the following conditions are met:

a) All implemented IPI API functions shall behave as specified by ISO/IEC 12087-2;

b) All implemented IPI IPI-IIF capabilities shall be as specified by ISO/IEC 12087-3;

c) All implemented functions shall conform to at least one of the encodings or bindings specified in support of this Standard;

d) The implementation conforms to least one of the Profiles specified in ISO/IEC 12087;

e) A mechanism shall be provided that enables an application program to detect when extensions are being used.

8.2 Conformance of accuracy and precision

Minimum constraints are normally placed on the *accuracy* and *precision* of an implementation. The meanings of these two terms are:

accuracy — a measure of the fidelity with which a computed result compares with the mathematically correct result;

precision — a measure of the inherent quantization error of representation of a quantity.

Conformance of accuracy is addressed as a documentation requirement in Conformance Profiles. Conformance of precision is discussed in ISO/IEC 12087-2 and ISO/IEC 12087-3.

8.3 Extensions

Extensions are defined as those data types, API functions, and IPI-IIF capabilities that are not part of the base standard. An implementation that satisfies the conformance constraints and provides additional, non-standard functionality shall satisfy the following criteria to be a conforming implementation:

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

4851903 0607787 172

a) Provide an enquiry mechanism for the detection of extensions;

b) Follow the rules and guidelines for this Standard and its corresponding language bindings and data encodings;

c) Clearly specify the extensions and document them as extensions;

d) Ensure that the extensions do not cause Standard-conforming application programs to work incorrectly.

Functions described in this Standard that are supplied by an implementation in addition to those required by a particular application profile are considered as extensions to that profile. Mechanisms for detecting the existence of extensions are described in ISO/IEC 12087-2 and ISO/IEC 12087-3.

8.4 Conformance profiles

Profiles are a means of standardizing IPI implementations that have different functional scopes, in order to suit a variety of users. Profiles represent a minimum level of capability that an IPI implementation shall provide, and a maximum level of capability that an application program shall utilize. An implementation may exceed the minimum requirements of a given profile. In this way, an IPI implementation may support multiple profiles, each of which reflects a subset of the capabilities implemented.

Each profile contains sufficient function support and resources for it to be used effectively for its intended purpose independently of other profiles. Given any two profiles, a single API or IPI-IIF implementation may be provided that satisfies both profiles. This implies that a profile may not be defined with any restrictions or assumptions that could prevent this 'profile compatibility' principle.

In general, profiles may impose requirements on resource availability and functional capacity that may be interpreted as demands on a memory resource. For implementations that use dynamic memory allocation, the interpretation of the profile may be taken as to mean the total memory needed to satisfy all such requirements. In this way, the same fixed memory resource may be allocated in different manners to satisfy different sets of requirements for different profiles.

This clause explains the principle of the definition of profiles. The details of the profiles defined by IPI and their content are defined in ISO/IEC 12087-2 and ISO/IEC 12087-3.

8.4.1 Types of profile

Three types of profile are defined, the *Foundation Profile*, the *Application Profile* and the *Full Profile*.

Foundation Profiles — there are two Foundation Profiles, one for IPI-PIKS and one for IPI-IIF. These specify a minimally-conforming level of compatibility to IPI-PIKS and IPI-IIF respectively.

Application Profiles — serve users who may require different classes of functionality from ISO/IEC 12087. Such profiles consist of a list of particular IPI-PIKS functions and a level of IPI-IIF support. All Application Profiles, either included in the ISO/IEC 12087 or subsequently registered shall be supersets of either the IPI-PIKS Foundation Profile or the IPI-IIF Foundation Profile. This means that an Application Profile shall include support for all of a Foundation Profile's required functions and interchange capabilities and meets the resource requirements of that Foundation Profile.

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

4851903 0607788 009

Full Profiles — include all permissible functionality available within ISO/IEC 12087. There are two Full Profiles, one for IPI-PIKS and one for IPI-IIF. A Full Profile is a superset of all Application Profiles.

The following information is contained in the definition of a profile:

— A name that is used to uniquely identify the profile, whether the profile is defined in ISO/IEC 12087 or is subsequently registered.

— A list of API functions that are required to be present. Such a list shall be complete in the sense that an implementation of only those functions may be used effectively for its intended purpose. It shall not contain functions that are useless in the absence of any function that is not listed.

— A statement of the minimum support requirements. Note that these minimum values represent the maximum capability that an application program may expect to be available.

Subject to approval by ISO, constituencies may impose other requirements in profiles, providing that such requirements do not violate any of the rules for profiles stated in ISO/IEC 12087. Where latitude is permitted in the behaviour of a function or capability, there is a preferred behaviour; this preferred behaviour is stated in ISO/IEC 12087. A profile shall not require a behaviour that conflicts with the preferred behaviour. A profile may require a particular behaviour even if ISO/IEC 12087 indicates that the behaviour is implementation-dependent.

8.4.2 Application profile registration

The profiles included in ISO/IEC 12087 are not expected to satisfy all future needs. Other profiles may be registered, when required, in the ISO Register of Graphical Items [ISO/IEC 9973:1994] or via International Standardized Profiles.

8.4.3 Profiles defined by IPI

ISO/IEC 12087 defines several profiles for both IPI-PIKS and IPI-IIF. Table 3 shows those profiles for IPI-IIF that correspond to IPI-PIKS profiles, while table 4 shows those IPI-PIKS profiles that correspond to the IPI-IIF profiles.

IPI-PIKS PROFILE	CORRESPONDING IPI-IIF PROFILE
IPI-PIKS Foundation	IPI-IIF Foundation
IPI-PIKS Technical	IPI-IIF Full IPI-PIKS
IPI-PIKS Scientific	IPI-IIF Full IPI-PIKS
IPI-PIKS Full	IPI-IIF Full IPI-PIKS

Table 3 — IPI-IIF profiles that correspond to IPI-PIKS profiles

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

4851903 0607789 T45

IPI-IIF PROFILE	CORRESPONDING IPI-PIKS PROFILE
IPI-IIF Foundation	IPI-PIKS Foundation
IPI-IIF Full IPI-PIKS	IPI-PIKS Technical, Scientific, Full
IPI-IIF Full	none

Table 4 — IPI-PIKS profiles that correspond to IPI-IIF profiles

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

4851903 0607790 767

Annex A

(normative)

Structured image data types

The purpose of this annex is to indicate the way in which familiar classes of images fit into the IPI-PIKS 5D image representation. The following paragraphs give textual descriptions of these image classes. Case is significant in the codes defined below.

Monochrome (MON) — Two-dimensional structure of pixels of the same data type with X columns and Y rows. A bi-tonal image is a special case of a monochrome image; it possesses only two grey level states.

Colour (COLR) — Three-dimensional structure of pixels of the same data type with X columns, Y rows, and three or four colour bands. Additive colour images have three colour bands and subtractive colour images have four colour bands. The registered colour representations are described in annex 8.4.3.

Volume (VOL) — Three-dimensional structure of pixels of the same data type with X columns, Y rows, and Z depth values.

Spectral (SPE) — Three-dimensional structure of pixels of the same data type with X columns, Y rows, and B spectral bands.

Temporal (TEM) — Three-dimensional structure of pixels of the same data type with X columns, Y rows, and T temporal values.

Volume-Colour (VC) — Four-dimensional structure of pixels of the same data type with X columns, Y rows, Z depth values, and three or four colour bands. The supported colour bands are the same as for a colour image.

Volume-Spectral (VS) — Four-dimensional structure of pixels of the same data type with X columns, Y rows, Z depth values, and B spectral bands.

Volume-Temporal (VT) — Four-dimensional structure of pixels of the same data type with X columns, Y rows, Z depth values, and T temporal values.

Temporal-Colour (TC) — Four-dimensional structure of pixels of the same data type with X columns, Y rows, T temporal values, and three or four colour bands. The supported colour bands are the same as for a colour image.

Temporal-Spectral (TS) — Four-dimensional structure of pixels of the same data type with X columns, Y rows, T temporal values, and B spectral bands.

Volume-Temporal-Colour (VTC) — Five-dimensional structure of pixels of the same data type with X columns, Y rows, Z depth values, T temporal values, and three or four colour bands. The supported colour bands are the same as for a colour image.

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

4851903 0607791 6T3

Volume-Temporal-Spectral (VTS) — Five-dimensional structure of pixels of the same data type with X columns, Y rows, Z depth values, T temporal values, and B spectral bands.

Generic (GEN) — Five-dimensional structure of pixels of the same data type with dimensions of (X, Y, Z, T, B). No interpretation is placed upon the semantics of the five dimensions by ISO/IEC 12087.

NOTES

19 It is possible for different images, having different attributes, to refer to the same pixel data. This makes it possible for an image to be considered as being of different structured data types according to the requirements of different operators. This is accomplished in IPI-PIKS by means of, for example, an 'index reassignment' mechanism.

20 Some IPI-PIKS operators only provide functionality for a subset of the image structural types, as defined in each operator definition. Some operators are three-dimensional in nature, *e.g.* temporal sequence average. Many operators are inherently two-dimensional, *e.g.* unsharp masking. When applied to multi-band images, such operators process each depth, temporal, colour or spectral band independently.

This same information may be conveniently summarized in the following table. The values given in the DIMENSIONS column refer to the numbers of possible subscript values for each of the $\{x, y, z, t, b\}$ dimensions.

MNEMONIC	MEANING	DIMENSIONS
MON	monochrome	$X, Y, 1, 1, 1$
COLR	colour	$X, Y, 1, 1, B$
VOL	volume	$X, Y, Z, 1, 1$
SPE	spectral	$X, Y, 1, 1, B$
TEM	temporal	$X, Y, 1, T, 1$
VC	volume-colour	$X, Y, Z, 1, B$
VS	volume-spectral	$X, Y, Z, 1, B$
VT	volume-temporal	$X, Y, Z, T, 1$
TC	temporal-colour	$X, Y, 1, T, B$
TS	temporal-spectral	$X, Y, 1, T, B$
VTC	volume-temporal-colour	X, Y, Z, T, B
VTS	volume-temporal-spectral	X, Y, Z, T, B
GEN	generic	X, Y, Z, T, B

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

4851903 0607792 53T

Annex B

(normative)

Structure codes

In clause 5 and clause 6, mention is made of various 'structure codes'. These are values that are used within IPI-PIKS or IPI-IIF or passed between IPI-PIKS and IPI-IIF to indicate the presence of various pieces of information. This annex, which is normative, summarizes these codes and their meanings.

NOTE 21 Character codes have been chosen in preference to integers because of their mnemonic value.

MNEMONIC	MEANING	MNEMONIC	MEANING
'AS'	application-specific	'PL'	power spectrum label
'BC'	Boolean data type	'QD'	quantization depth
'BL'	bi-level	'RC'	real data type
'CC'	complex data type	'SC'	signed integer data type
'CO'	colour	'SL'	structuring element label
'DL'	dither label	'ST'	scan type
'FE'	feature	'TC'	character string data type
'GC'	gamma correction	'TL'	transfer function label
'GL'	generic label	'TP'	transparency
'GV'	grey-value	'WL'	window function label
'HT'	halftone	'1D'	one-dimensional
'ID'	internal identifier	'2D'	two-dimensional
'IP'	external identifier	'3D'	three-dimensional
'IL'	impulse response label	'4D'	four-dimensional
'ML'	mask label	'5D'	five-dimensional
'NC'	non-negative integer data 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 0607793 476

Annex C

(normative)

The representation of colour

Image processing is a visual science and colour plays an important in the visual process; hence the representation of colour is an important aspect of ISO/IEC 12087. Colour is a perceptual concept, not a strictly objective or physical phenomenon. Nevertheless, to permit the manipulation of colour data, various representations have been developed. This annex, which is normative, describes those colour representations that are supported by ISO/IEC 12087.

In discussing colour in relation to IPI, it is desirable to differentiate between the concepts of *colour systems* and *colour spaces*:

colour system — a particular representation of colour that is independent of all other representations; it is *not* possible to convert from one colour system to another in a fully colour-preserving way.

NOTE 22 However, within a given context, such as fixed printing and viewing conditions, it may be possible to perform a conversion that will preserve colours.

colour space — a specific instance of a colour system; colours represented within a particular colour space may be converted to other colour spaces that describe the same colour system.

Furthermore, the term *colour representation* means, in the context of this Standard, a colour space associated with a specific colour system. The distinction between colour systems and colour spaces, which is illustrated in figure 8, will become more apparent in the following paragraphs.

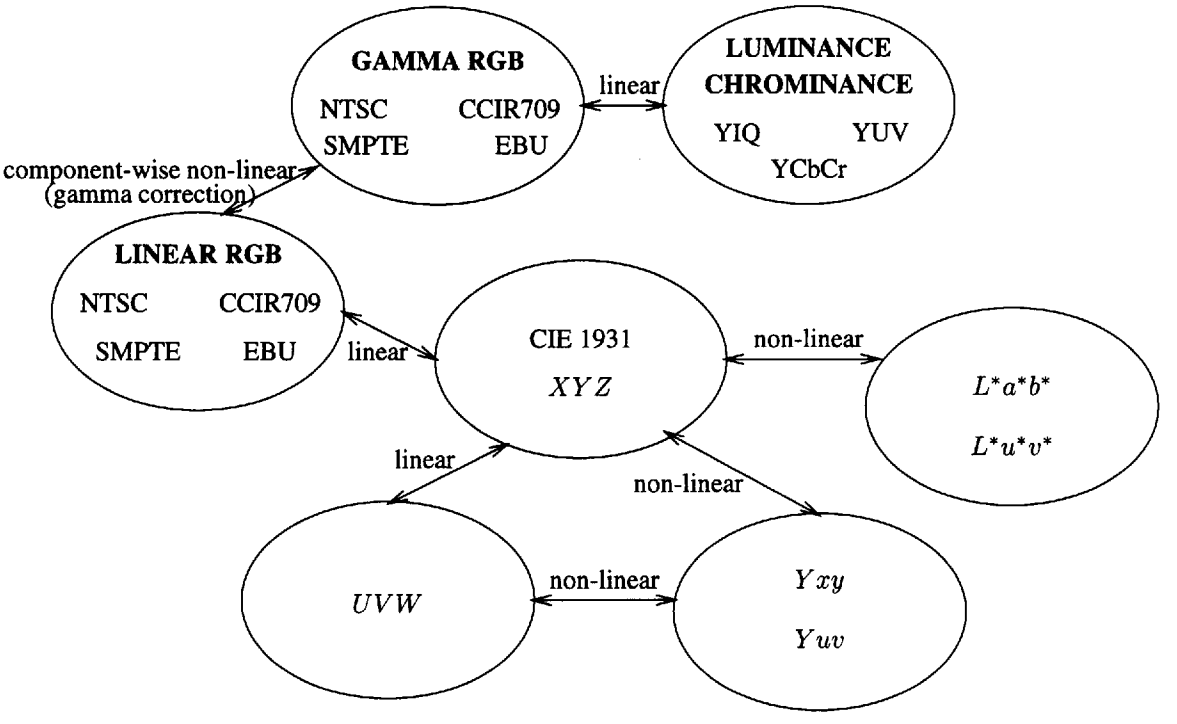
NOTE 23 The linear transformations between colour spaces indicated in figure 8 may be accomplished by means of matrix arithmetic.

IPI identifies the following colour systems:

CIE 1931 — the principal colour system used in image processing and computer graphics, defined for an observer with a two-degree field of view; the fundamental space in this system is (X, Y, Z) [CIE:1931].

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

4851903 0607794 302



standardized

non-standardized

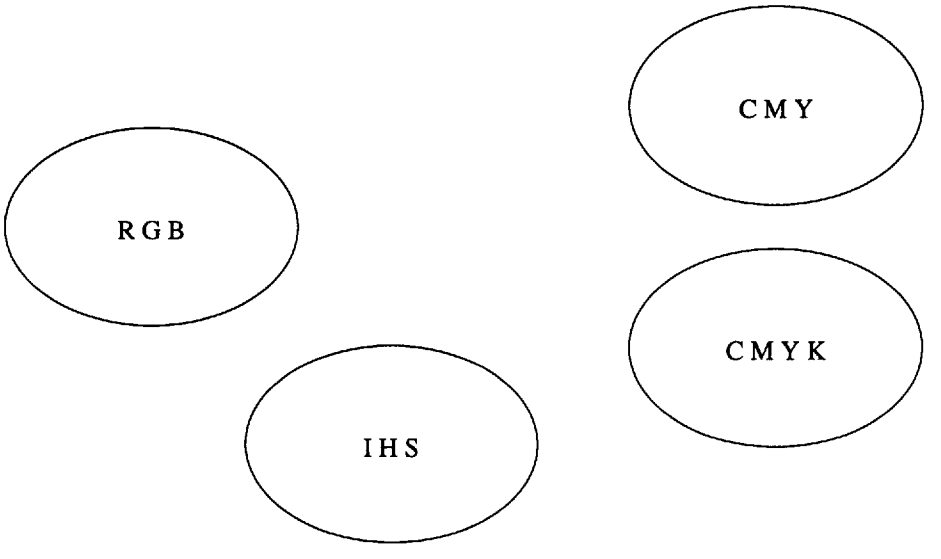


Figure 8 — Colour Systems and Representations Used by IPI

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

4851903 0607795 249

ILLUMINANT	X_0	Y_0	Z_0
A	1.098700	1.000000	0.355900
C	0.980708	1.000000	1.182163
$D50$	0.964296	1.000000	0.825105
$D65$	0.950456	1.000000	1.089058
E	1.000000	1.000000	1.000000

Table 5 — XYZ tristimulus values for the white points of common illuminance

CIE 1964 — an alternative to the CIE 1931 system, defined for an observer with a ten-degree field of view; the fundamental space in this system is (X_{10}, Y_{10}, Z_{10}) [CIE:1976].

NOTE 24 This system is not currently supported in ISO/IEC 12087.

non-standardized RGB, CMY, CMYK, *etc.* systems — commonly used for images derived from non-CIE-calibrated sources.

Within each colour system, several categories of representation are identified. Most of these categories have associated parameters in order to describe them fully. Table 6 shows which colour representation falls into which category and indicates the type of associated information that is required to produce a complete description. For example, to derive $L^*a^*b^*$ from XYZ, one must specify a white point. IPI permits these associated parameters to take on any valid values. Certain values for these parameters, corresponding to widely-used colour representations, are specifically supported and are assigned particular names. These are listed in table 7.

Additional parameterisations of colour representations will be supported by IPI if they are registered with the appropriate registration authority — see clause 8.

For those representations that have been developed for broadcast television and related applications (see table 7), certain illuminants (white points) are in common use. IPI provides explicit support for five commonly-used illuminants, the white-point (XYZ) tristimulus values of which are shown in table 5.

NOTE 25 The tristimulus values quoted in table 5, and the transformation matrices given later in this Annex, are quoted to six decimal places. In some cases, this is a greater number of places than their original specification. The number of places has been increased to avoid the introduction of spurious errors when concatenating transformations between colour representations.

Conversion between specific colour representations (for specific white points, in some cases) may frequently be represented as a matrix product. A matrix that converts one colour representation to another is termed a colour conversion matrix. Colour conversion matrices for common conversions are available in IPI-PIKS via its data object repository and in IPI-IIF. These matrices are derived from the definitions listed below.

CCIR709 —

	x	y	z
red	0.640000	0.330000	0.030000
green	0.300000	0.600000	0.100000
blue	0.150000	0.060000	0.790000

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

4851903 0607796 185

CIE —

	x	y	z
red	0.734690	0.265310	0.000000
green	0.273680	0.717430	0.008900
blue	0.166540	0.008880	0.824580

EBU —

	x	y	z
red	0.640000	0.330000	0.030000
green	0.290000	0.600000	0.110000
blue	0.150000	0.060000	0.790000

NTSC —

	x	y	z
red	0.670000	0.330000	0.000000
green	0.210000	0.710000	0.080000
blue	0.140000	0.080000	0.780000

SMPTE-C —

	x	y	z
red	0.630000	0.340000	0.030000
green	0.310000	0.595000	0.095000
blue	0.155000	0.070000	0.775000

Within IPI-PIKS, colour images are represented as multi-band images that have an associated colour attribute (see clause 5). There is a unique mapping of colour bands onto bands for those representations that are supported; these mappings are tabulated in table 9. Further detail of the various colour representations are given in ISO/IEC 12087-2.

NOTES

26 IPI-IIF transports both the names and the parameter values — both are useful, and providing both ensures that IPI-IIF-encoded images can still be read correctly even if the colour standards are revised.

27 Additional categories of representation are not permitted within this Standard, so that all implementations of IPI-IIF will be able to handle all colours, even if more are registered. New categories of representation may be included only by amending this Standard.

28 The 'OTHER ATTRIBUTES' column of table 6 indicates whether additional information is required. If this is the case, such information (*e.g.*, manufacturer and model of scanner) should be supplied as a character string. Such information is *not* interpreted by the IPI 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 0607797 011

REPRESENTATION	REFERENCE	PRIMARIES	WHITE POINT	TRANSFER FUNCTION	LUMINANCE COEFFICIENTS	OTHER ATTRIBUTES
CIE XYZ	[CIE:1931]	no	no	no	no	no
CIE Yxy	[CIE:1931]	no	no	no	no	no
CIE UVW	[CIE:1970]	no	no	no	no	no
CIE Yuv	[CIE:1970]	no	no	no	no	no
CIE $L^*a^*b^*$	[CIE:1976]	no	yes	no	no	no
CIE $L^*u^*v^*$	[CIE:1976]	no	yes	no	no	no
Linear RGB		yes	yes	no	no	no
Gamma RGB		yes	yes	yes	no	no
Luminance-chrominance		yes	yes	yes	yes	no
non-standardized RGB		no	no	no	no	yes
non-standardized CMY		no	no	no	no	yes
non-standardized CMYK		no	no	no	no	yes
non-standardized IHS		no	no	no	no	yes

Table 6 — Supported types of colour representation, and their attributes

REPRESENTATION	REFERENCE
Linear RGB	
linear NTSC RGB	[CCIR:1990a], [CCIR:1990b]
linear EBU RGB	[CCIR:1990a], [CCIR:1990b], [EBU:1975]
linear SMPTE RGB	[DeMarsh:1974]
linear CCIR-709 RGB	[CCIR:1990c]
Gamma RGB	
gamma NTSC RGB	[CCIR:1990b]
gamma EBU RGB	[CCIR:1990b]
gamma SMPTE RGB	[DeMarsh:1974]
gamma CCIR-709 RGB	[CCIR:1990c]
Luminance-Chrominance	
YIQ	[NTSC:1954]
YUV	[CCIR:1990b]
SMPTE YC_bC_r	[DeMarsh:1974]

Table 7 — Standardized parameterisations of colours

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

4851903 0607798 T58

REPRESENTATION	PRIMARIES x, y	WHITE POINT	TRANSFER FUNCTION	LUMINANCE COEFFICIENTS
CIE representations				
XYZ	no	no	no	no
Yxy	no	no	no	no
UVW	no	no	no	no
Yuv	no	no	no	no
$L^*a^*b^*$	no	yes	no	no
$L^*u^*v^*$	no	yes	no	no
Linear RGB				
linear NTSC RGB	yes	yes	no	no
linear EBU RGB	yes	yes	no	no
linear SMPTE RGB	yes	yes	no	no
linear CCIR-709 RGB	yes	yes	no	no
Gamma RGB				
gamma NTSC RGB	yes	yes	yes	no
gamma EBU RGB	yes	yes	yes	no
gamma SMPTE RGB	yes	yes	yes	no
gamma CCIR-709 RGB	yes	yes	yes	no
Luminance-Chrominance				
YIQ	yes	yes	yes	yes
YUV	yes	yes	yes	yes
SMPTE YC_bC_r	yes	yes	yes	yes

Table 8 — Parameter values for the standardized colour representations (non-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

4851903 0607799 994

COLOUR SPACE	CHANNEL 1	CHANNEL 2	CHANNEL 3	CHANNEL 4
CIE XYZ	X	Y	Z	null
CIE $L^*a^*b^*$	L^*	a^*	b^*	null
CIE $L^*u^*v^*$	L^*	u^*	v^*	null
CIE UVW	U	V	W	null
CIE Yxy	Y	x	y	null
CIE Yuv	Y	u	v	null
CMY	C	M	Y	null
CMYK	C	M	Y	K
gamma NTSC RGB	R	G	B	null
gamma EBU RGB	R	G	B	null
gamma SMPTE RGB	R	G	B	null
gamma CCIR RGB	R	G	B	null
IHS	I	H	S	null
linear CIE RGB	R	G	B	null
linear NTSC RGB	R	G	B	null
linear EBU RGB	R	G	B	null
linear SMPTE RGB	R	G	B	null
linear CCIR RGB	R	G	B	null
uncalibrated RGB	R	G	B	null
luminance-chrominance YIQ	Y	I	Q	null
luminance-chrominance YUV	Y	U	V	null
luminance-chrominance YC_bC_r	Y	C_r	C_b	null

Table 9 — Mappings Between Colours and Image Channels

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

4851903 0607800 436

Annex D

(normative)

Language-Independent Data Types

D.1 Bit

Description — Bit is the numeric data type representing the finite field of two symbols designated "0", the additive identity, and "1", the multiplicative identity.

Syntax — Bit = "bit".
bit-value = "0" | "1".

Values — 0, 1.

Properties — numeric, ordered, discrete, bounded.

Operations — InOrder, Add, Multiply.

InOrder(x, y:bit):Boolean, Add(x, y:bit):bit, Multiply(x, y:bit):bit are defined by table, as follows:

x	y	Add (x, y)	Multiply (x, y)	InOrder(x, y)
1	1	0	1	true
1	0	1	0	false
0	1	1	0	true
0	0	0	0	true

D.2 Boolean

Description — Boolean is the mathematical data type associated with two-valued logic.

Syntax — Boolean = "Boolean".
Boolean-value = "true" | "false".

Values — "true", "false", such that true $\neq$ false.

Properties — non-numeric, unordered, discrete.

Operations — Not, And, Or.

Not(x:Boolean):Boolean is defined by tabulation:

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

4851903 0607801 372

x	Not(x)
true	false
false	true

Or(x, y:Boolean):Boolean is defined by tabulation:

x	y	Or(x, y)
true	true	true
true	false	true
false	true	true
false	false	false

And(x, y:Boolean):Boolean = Not(Or(Not(x),Not(y))).

NOTE 29 Either And or Or is sufficient to characterize the Boolean data type, and given one, the other can be defined in terms of it. They are both defined here because both of them are used in the definitions of operations on other data types.

D.3 Character

Description — Character is a family of data types, each of which represents the symbols in a registered alphabet.

Syntax — Character = "character" "(" parameter ")".
character-value = literal — positional-value.
positional-value = alphabet-identifier "." number.
alphabet-identifier = identifier.

Parameters — When the parameter is a value, it shall be an alphabet-identifier. The parameter shall be a value, except for some occurrences in declarations. An *alphabet-identifier* is effectively a value of a State data type, defined by another standard, which associates each such identifier with an ordered character-set. The character sets themselves are defined by further standards.

Values — Each Character data type has a finite value space whose members are character-values in the character-set identified by alphabet-identifier, each of which is distinct from all the others, and whose ordering is given by the character-set. A *literal* value denotes that symbol in the character-set identified by alphabet-identifier which is identified with the character *any-character* from the character-set. A *positional-value* denotes the symbol which occupies the *number*th position in the character-set by alphabet-identifier.

Properties — non-numeric, ordered, discrete, bounded.

Operations — InOrder, Successor.

Because Character is a family of data types, the occurrences of Character in the following operation definitions must be uniformly replaced by the designator for a given Character data type in order to define the operations for that data 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 0607802 209

InOrder(x, y:Character):Boolean, denoted $x \leq y$, is true if $x = y$ or if x precedes y in the alphabet, else false.

Successor(x:Character):Character is if for all y: Character, $x \leq y$ implies $x = y$, then undefined; else the value y: Character, such that $y \neq x$ and for all $z \neq x$, $x \leq z$ implies $y \leq z$.

NOTES

30 The Character data types are distinct from the Enumerated data types in that the values of the data type and their ordering are defined by other standards rather than by the application. This distinction is semantically unimportant, but it is of great significance in any use of these standards.

31 While ordering is important in many applications of Character data types, there will also be applications in which the ordering is unimportant. Such applications can either select the identifier for a standard ordering (and never use the ordering) or define a new State data type with the character values as its values.

D.4 Complex

Description — Complex is the mathematical data type comprising the "complex numbers."

Syntax — Complex = "complex"
complex-value = "(" r-part "," i-part ")".
r-part = real-part.
i-part = real-part.

Values — The value space of Complex is the mathematical field created by the algebraic closure of the real values, *i.e.* the solution space of all polynomial equations having real coefficients. There are two isomorphic value spaces for Complex which differ only in which of the two square-roots of -1 is taken to be the "principal square root", designated i. (The other square-root is $-i$.) Any complex value then has a unique representation of the form: $x + y \times i$, where x and y are real-values, designated the *real* part and the *imaginary* part respectively. Accordingly, in the denotation of a complex-value (*r-part*, *i-part*) denotes: Add(Promote(*r-part*), SquareRoot(Negative(Multiply(Promote(*i-part*),Promote(*i-part*))))).

Properties — numeric, dense, unordered.

Operations — Promote, Negate, Add, Multiply, Reciprocal, SquareRoot, Measure

Promote(x:real) — complex is the embedding isomorphism.

Add(x, y:complex) — complex is the additive operation on the field.

Multiply(x, y:complex) — complex is the multiplicative operation on the field.

Negate(x:complex) — complex produces the additive inverse of x.

Reciprocal(x:complex) — complex produces the multiplicative inverse of x.

SquareRoot(x:complex) — complex is the value y: complex such that Multiply(y, y) = x, and either the real-part of y is zero and the imaginary part is non-negative, or the real-part of y is positive.

Measure(x, y:complex) — real is ???

D.5 Enumerated

Description — Enumerated is a family of data types, each of which comprises a finite number of distinguished values having an intrinsic ordering.

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

4851903 0607803 145

Syntax — Enumerated = "enumerated" "(" parameter-list ")".
enumerated-value = value-identifier.

Parameters — When the parameters in the parameter-list are values, they shall be enumerated-values. The parameters shall be values, except for some occurrences in declarations.

Values — Each Enumerated data type has a finite set of named values, each of which is designated by a unique *value-identifier* and is distinct from all other values of the data type. The ordering of these values is given by the sequence of their occurrence in the parameter-list, designated the *naming sequence*.

Properties — non-numeric, ordered, discrete, bounded.

Operations — InOrder, Successor.

Because Enumerated is a family of data types, the occurrences of Enumerated in the following operation definitions must be uniformly replaced by the designator for a given enumerated data type in order to define the operations for that data type.

InOrder(x, y:Enumerated):Boolean, denoted $x \leq y$, is true if $x = y$ or if x precedes y in the naming sequence, else false.

Successor(x:Enumerated):Enumerated is if for all y:Enumerated, $x \leq y$ implies $x = y$, then undefined; else the value y:Enumerated, such that $x \neq y$ and for all $z \neq x$, $x \leq z$ implies $y \leq z$.

D.6 Null

Description — Null is the data type representing an object whose presence is syntactically or semantically required, but has no value (by intention) in a given instance.

Syntax — Null = "null".
null-value = "*".

Values — a single value, designated the null-value.

Properties — non-numeric, unordered, discrete.

Operations — only evidence: any two objects of data type Null are equal.

NOTES

32 Data Type Null is defined for use in Choice constructions to represent optional objects or variants of an information structure which have no content.

33 The "empty set" is not a value of data type Null, but rather a value of the appropriate Set-generated data type.

D.7 Integer

Description — Integer is the mathematical data type comprising the exact integral values.

Syntax — Integer = "integer".
integer-value = signed-number.
signed-number = ["–"] 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 0607804 081

Values — Mathematically, the infinite ring produced from the additive identity (0) and the multiplicative identity (1) by requiring Add(x,1) $\neq y$ for an $y \leq x$. That is: $\ldots -2, -1, 0, 1, 2, \ldots$(a denumerably infinite list).

Properties — numeric, ordered, discrete, unbounded.

Operations — InOrder, NonNegative, Negate, Add, Multiply, Promote

Promote(x:bit) — integer is defined by tabulation:

x	Promote(x)
0	0
1	1

Add(x, y:integer) — integer is the additive operation on the ring.

Multiply(x, y:integer) — integer is the multiplicative operation on the ring.

Negate(x:integer) — integer produces the additive inverse of x.

NonNegative(x:integer) — Boolean is true if $x = 0$ or x can be developed by one or more iterations of adding 1, *i.e.* if x = Add(1, Add(1, ...Add(1, Add(1,0)) ...)); else false.

InOrder(x, y:integer) — Boolean = NonNegative(Add(x, Negate(y))).

D.8 Real

Description — Real is the mathematical data type comprising the "real numbers".

Syntax — Real = "real".
real-value = signed-number ["." number].

Values — The value space of Real is the mathematical field comprising all values which are limits of convergent sequences of rational numbers.

Properties — numeric, ordered, dense, unbounded.

Operations — InOrder, Promote, Negate, Add, Multiply, Reciprocal, Measure.

Promote(x:rational) — real is the embedded homomorphism.

Add(x, y:real) — real is the additive operation on the field.

Multiply(x, y:real) — real is the multiplicative operation on the field.

Negate(x:real) — real produces the additive inverse of x.

Reciprocal(x:real) — real produces the multiplicative inverse of x.

NonNegative(v:real) — Boolean is defined by: For every real value v, there is a convergent sequence of rational values r_i such that limit(r_i)= v, and there is some k such that for all $i > k$, NonNegative(r_i) = NonNegative(r_k). Then define: NonNegative(v) = rational.NonNegative(r_k).

InOrder(x, y:rational) — Boolean = NonNegative(Add(x, Negate(y))).

Measure(x, y:real) — real is defined by: if InOrder(x, y) then Add(y, Negate(x)), else Add(x, Negate(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 0607805 T18

NOTE 34 The common language-independent data type Real is the abstract mathematical data type. The relationship of this data type to the common "floating-point" implementations must be defined for each implementation by the use of "pragmata." For languages whose semantics in some way assumes a floating-point representation, the use of such pragmata in the data type mappings may also be necessary. On the other hand, for some usages, the machine representation of the Real data type may be something other than floating-point, which then only requires different pragmata.

D.9 State

Description — State is a family of data types, each of which comprises a finite number of distinguished but unordered values with no characterizing operations.

Syntax — State = "state" "(" parameter-list ")".
state-value = value-identifier.

Parameters — When the parameters in the parameter-list are values, they shall be state-values. The parameters shall be values, except for some occurrences in declarations.

Values — Each State data type has a finite set of named values, each of which is designated by a unique *value-identifier* and is distinct from all other values of the data type.

Properties — non-numeric, unordered, discrete.

Operations — only equivalence.

D.10 Array

Description — Array creates a new data type whose valueds are fixed-length sequences of values from the element data type, such that each value in the sequence associates to a unique value of another data type, designated the *index* data type, and each value of the index data type associates to some value of the element data type.

Syntax — Array-type = "array" "(" index ")" "of" element.
index = data type. element = data type.

Components — two data types. The first, designated the *index* data type, must be a bounded, discrete data type; and the second, designated the *element* data type, may be any data type.

Values — all sequences of values from the element data type having a length equal to the cardinality of the index data type.

Properties — non-numeric, unordered, discrete.

Subtypes — any array-type having the same index data type as the base data type and an element data type which is a subtype of the base element data type.

Operations — Equal, Select, Replace.

Let *lowerbound* and *upperbound* be the lower and upper bounds on the value space of data type *index*. In order to define the operations on the Array data type, it is first necessary to define a mapping from the index data type to the Ordinal data 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 0607806 954

Ord(x:index) — Ordinal is defined by:

Ord(*lowercase*) — is *first*; and

Ord(Successor(x)) — is Successor(Ord(x)), for all $x \neq$ *upperbound*.

Then the proper operations on the Array data types are defined as follows:

Select(x:array(*index*) of *element*, y:*index*):*element* — yields that value of the element data type which is the Ord(y) element of the sequence x;

Equal(x, y:array(*index*) of *element*):Boolean — is true if for every value v in the value space of data type *index*, Select(x, v) = Select(y, v), else false.

Replace(x:array(*index*) of *element*, y:*index*, z:*element*):array(*index*) of *element* — yields a new value of the array type which is obtained by substitution of the value z for the Ord(y) element of the sequence x.

NOTE 35 A "multidimensional array" is conceptually identical to recursive application of the Array generator. For example, the Pascal declaration:

```
var screen : array [1..24, 1..80] of char;
```

declares the variable "screen" to have the common language-independent data type:

```
array (range (1:24) of integer) of
  array (range (1:80) of integer) of character (local).
```

And the Pascal subscript operation:

```
screen[row, column]
```

is equivalent to the characterizing operations:

```
Select (Select (screen, row), column).
```

D.11 Choice

Description — Choice creates a new data type each of whose values is a single value from one of a set of alternative data types.

Syntax — Choice-type = "choice" "of" alternative-list.
alternative-list = "any" — "(" data type-list ")".
data type-list = data type "," data type .

Components — A set of alternative data types, each of which may be any data type. When the Any form of the *alternative-list* is used, it indicates that the set of alternatives comprises all possible data types.

Values — all values from the spaces of all of the alternative data types.

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

4851903 0607807 890

Properties — non-numeric, unordered.

Subtypes — any single data type in the alternative-list, or a subtype thereof, or a Choicetype in which every data type in the alternative-list of the subtype appears in, or is a subtype of a data type in, the alternative-list of the base data type.

Operations — Equal, Label, Cast, IsType

IsType.*alternative*(x: choice of *alternative-list*) — Boolean, where *alternative* is a data type in *alternative-list*, is true if x is tagged as "derived from" the *alternative* data type, and false otherwise.

Label(x: *alternative*) — choice of *alternative-list*, where *alternative* is a data type in *alternative-list*, produces that value of the Choice data type which is identical to the value x, logically tagged as "derived from" the *alternative* data type.

Cast_*alternative*(x: choice of *alternative-list*) — *alternative*, where *alternative* is a data type in *alternative-list*, yields:
if IsType.*alternative*(x), then that value of the *alternative* data type which is identical to x,
else, undefined.

Equal(x, y: choice of *alternative-list*) — Boolean is:
if there exists an *alt* data type in *alternative-list* such that And(IsType.*alt*(x), IsType.*alt*(y)),
then
Equal.*alt*(Cast.*alt*(x), Cast.*alt*(y)),
where Equal.*alt* is the equivalence relationship on data type *alt*,
else false.

NOTES

36 The Any form of *alternative-list* is provided for those cases in which all of the alternative data types which may appear in place of the generator cannot be known *a priori*. For example, consider the case in which a Record data type comprises several fields which describe an information unit and one field which is an instance of the information unit. The data types of the descriptive fields are known, but the data type of the information unit field must be "choice of any". In this case, the Any stands for all of those data types which can be described by the descriptive fields.

37 The Choice data type-generator is referred to in some programming languages as a "union" data type, and in others as a data type with "variants". Choice of Any is sometimes called "generic".

38 The subtypes of a Choice-type are typically Choice-types with a smaller list of alternatives, and in the simplest case, the list is reduced to a single data type.

D.12 List

Description — List creates a new data type whose values are ordered sequences of values from the element data type. The ordering is imposed on the values and not intrinsic in the underlying data type; the same value may occur more than once in a given sequence.

Syntax — List-type = "list" ["(" limits ")"] "of" element.
limits = minimum-size ["," maximum-size].
minimum-size = parameter.
maximum-size = parameter.
element = data 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 0607808 727

Components — any single data type, designated the *element* data type. The optional *limits* specify the minimum and maximum number of elements in the list values. If the minimum-size is omitted, it is taken to be zero, and if the maximum-size is omitted, it is taken to be unlimited. When minimum-size and maximum-size are values, they shall be non-negative integer values, and if both are given, then minimum-size shall be lees than or equal to maximum-size. Minimum-size and maximum-size shall be values except in some occurrences in declarations.

Values — all finite sequences of values from the element data type, which contain at least *minimum-size* values and at most *maximum-size* values.

Properties — non-numeric, unordered, discrete.

Subtypes — any list-type in which the element data type of the subtype is the same as, or a subtype of, the element data type of the base list-type, and the limits on the subtype are such that base-minimum-size $\geq$ subtype-minimum-size, and subtype-minimum-size $\geq$ base-maximum-size.

Operations — IsEmpty, Head, Tail, Equal, Empty, Append.

IsEmpty(x: list of *elements*): Boolean — is true if the sequence x contains no values, else fails;

Head(x: list of *element*): *element*, — where Not(IsEmpty(x)), is the first value in the sequence x;

Tail(x: list of *element*): list of *element* — is the sequence of values formed by deleting the first value from the sequence x, if any;

Equal(x, y: list of *element*): Boolean is —
if IsEmpty(x), then IsEmpty(y);
else if Head(x) = Head(y), then Equal(Tail(x), Tail(y));
else, false;

Empty(): list of *element* — is a niladic operation yielding the sequence containing no values of data type *element*;

Appendix(x: list of *element*, y: *element*): list of *element* — is the sequence formed by adding the single value y to the end of the sequence x.

NOTE 39 The notion *sequential file*, meaning "a sequence of values of a given data type, usually stored on some external medium", is an implementation of data type List.

D.13 Pointer

Description — Pointer creates a new data type each of whose values is a means of reference to an instance of a value of the element data type.

Syntax — Pointer-type = "pointer" "to" element.
element = data type.

Components — Any single data type, designated the *element* data type.

Values — The value space is that of an unspecified State data type, each of whose values identifies an instance of a value of the element data type.

Properties — non-numeric, unordered, discrete.

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

4851903 0607809 663

Subtypes — any pointer-type for which the element data type is a subtype of the element data type of the base pointer-type.

Operations — Equal, Dereference.

Equal(x, y: pointer to *element*): Boolean — is true if the values x and y identify the same instance of the element data type, else false;

Dereference(x: pointer to *element*): *element* — yields the value of the element data type possessed by the instance which the value of x identifies;

NOTE 40 Two pointer are equal only if they identify the same *instance* of the element data type; it does not suffice that they point to the same *value*. The operation which compares the values is Equal(Dereference(x), Dereference(y)).

D.14 Range

Description — Range creates a subtype of some ordered data type by placing new upper and/or lower bounds on the value space.

Syntax — Range-subtype = "range" "(" [lowerbound] "," [upperbound] ")" "of" base.
lowerbound = parameter.
upperbound = parameter.
base = data types.

Components — The *base* data type shall be ordered. Range is further parametrized by two values from the base data type, designated the *lowerbound* and *upperbound* of the subtype, of which either, but not both, may be omitted. When *lowerbound* and *upperbound* are values, they shall be values from the base data type such that InOrder(*lowerbound*, *upperbound*). *Lowerbound* and *upperbound* shall be values, except in some occurrences in declarations.

Values — all values v from the *base* data type such as *lowerbound* $\geq v$, if *lowerbound* is given, and $v \geq$ *upperbound*, if *upperbound* is given.

Properties — numeric if and only if the base data type is numeric, ordered, dense if and only if the base data type is dense, bounded (above, below, both) if the base data type is so bounded or the corresponding bound is given.

D.15 Record

Description — Record creates a new data type whose values are aggregations of values from a collection of component data types, each aggregation having one value for each component data type.

Syntax — Record-type = "record" of "(" field-list ")".
field-list = field "," field .
field = field-identifier ":" field-type.
field-identifier = identifier.

Components — A list of *fields*, each of which associates an identifier, designed a *field-identifier*, with a single data type, designated the *field-type*, which may be any data 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 0607810 385

Values — all collections of values, one per field in the argument list, such that the data type of each value is that of the field to which it corresponds.

Properties — non-numeric, unordered, discrete.

Subtypes — any Record-type with the same field-identifiers as the base data type, such that the field-type of each field of the subtype is the same as, or is a subtype of, the corresponding field-type of the base data type.

Operations — Equal, FieldSelect, Aggregate.

Equal(x, y: record of (*field-list*)): Boolean is — true if for every field-identifier f of the Record data type. Equal.*field-type*(FieldSelect.f(x), FieldSelect.f(y)), where in each case, Equal.*field-type* is the equivalence relationship on the field-type of the field with field-identifier f.

Aggregate(x: *field-type*$_1$, y: *field-type*$_2$ ...z: *field-type*$_n$) — record of (*field-list*) yields the value of the record data type whose first field has value x and whose second field has value y, *etc.*

There is one FieldSelect operation for each field in the Record-type. Each has the form:

FieldSelect.*field-identifier*(x: record of (*field-list*)): *field-type* — yields the value of the field of record x whose field-identifier is *field-identifier*.

NOTES

41 The sequence of fields in a Record data type is not semantically significant in the definition of the Record data type generator. That is, the use of the ordering to define the Aggregate operation is a purely syntactic convention which facilitates the definition of the operation. In practice any equivalent aggregating operation suffices. An implementation of a Record data type may define a representation convention which is an ordering of physically distinct fields, but that is a pragmatic consideration and not a part of the conceptual notion of the data type. Indeed, the optimal representation for certain Record values might be a bit-string, and then Aggregate would be an encoding operation and FieldSelect would be a decoding operation.

42 A record-type with a subset of the fields of a base record-type (a "sub-record" or "projection" of the record-type) is *not* a subtype of the base record-type: none of the values in the sub-record value space appears in the base value-space. And there are, in general, a great many different "embeddings" which map the sub-record data type into the base data type, each of which supplies different values for the missing fields. Supplying Null values for the missing fields is only possible if the data types of those fields are of the form Choice of (..., Null).

D.16 Set

Description — Set creates a new data type whose value-space consists of values from the powerset of the element data type, with operations appropriate to the mathematical *set*.

Syntax — Set-type = "set" ["(" limits ")"] "of" element.
limits = minimum-size ["," maximum-size]. minimum-size = parameter. maximum-size = parameter.
element = data type.

Components — Any discrete data type, designated the *element* data type. The optional *limits* specify the minimum and maximum number of elements in the set values. If the minimum-size is omitted, it is taken to be zero, and if the maximum-size is omitted, it is taken to be unlimited. When minimum-size and maximum-size are values, they shall be non-negative integer values, and if both are given, then minimum-size shall be less than or equal to maximum-size. Minimum-size and maximum-size shall be values except in some occurrences.

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

4851903 0607811 211

Values — every collection of distinct values from the value space of the *element* data type, containing at least *minimum-size* distinct values and at most *maximum-size* distinct values.

Properties — non-numeric, unordered, discrete.

Subtypes — any set-type in which the element data type of the subtype is the same as, or a subtype of, the element data type of the base set-type, and the limits on the subtype are such that base-minimum-size ≥ subtype-minimum-size, and subtype-maximum-size ≥ base-maximum-size.

Operations — IsIn, Subset, Equal, Complement, Union, Intersection, Empty, Setof, Select

IsIn(x: *element*), y: set of *element*): Boolean is — true if the value x appears in the collection y, else false;

Subset(x, y: set of *element*): Boolean is — true if for every value v of the *element* data type, IsIn(v, x) implies IsIn(v, x), else false;

Equal(x, y: set of *element*): Boolean is — And(Subset(x, y), Subset(y, x));

Complement(x: set of *element*): set of *element* — yields the set consisting of all values v from the value space of *element* such that Not(IsIn(v, x));

Union(x, y: set of *element*): set of *element* — yields the set consisting of all values v from the value space of *elements* such that Or(IsIn(v, x), IsIn(v, y));

Intersection(x, y: set of *element*): set of *element* — yields the set consisting of all values v from the value space of *elements* such that And(IsIn(v, y));

Empty(): set of *element* — is a niladic operation yielding the set which consists of no values from the value space of *element*;

Setof(y: *element*): set of *element* — yields the set consisting of the single value y;

Select(x: set of *element*): *element*, where Not(Equal(x, Empty()), — yields some one value space of *element* which appears in the collection x.

D.17 Character String

Description — Character-string is a family of data types which represent strings of symbols from registered alphabets.

Declaration — character-string (*alphabet*) = new list of character (*alphabet*);

Parameters — *alphabet* is an "alphabet-identifier".

Values — Each value of a character-string data type is a finite sequence of symbols from the given *alphabet*. The value-space comprises the collection of all such values.

Properties — non-numeric, ordered, discrete, bounded below, bounded above.

Operations — (First, Rest, Append, Length, Empty) from List, InOrder is an application of the collating sequence defined by the *alphabet* parameter.

NOTES

43 Character-string is assumed to be a List, rather than an Array, in that the values may be different lengths.

44 Applications which share a character set and a non-standard collating sequence need to create an alphabet-identifier which refers to the character-set and the agreed-upon collating 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 0607812 158

D.18 Table

Description — Table creates a new data type each of whose values is a collection of values from the cross-product of two data types, called the key data type and the element data type respectively.

Syntax — Table-type = "table" ["(" limits ")"] "of" "(" element "," key ")".
limits = minimum-size ["," maximum-size].
minimum-size = parameter.
maximum-size = parameter.
element = data type.
key = data type.

Components — two data types, designated the *key* data type and the *element* data type. The element data type may be any data type; and the key data type may be any discrete data type. The optional *limits* specify the minimum and maximum number of elements in the table values. If the minimum-size is omitted, it is taken to be zero, and if the maximum-size is omitted, it is taken to be unlimited. When minimum-size and maximum-size are values, they shall be non-negative integer values, and if both are given, then minimum-size shall be less than or equal to maximum-size. Minimum-size and maximum-size shall be values except in some occurrences in declarations.

Values — all finite sets of pairs of values, each pair having one value from the element data type and one value from the key data type, with the restriction that no value of the key data type can appear in more than one pair.

Properties — non-numeric, unordered, discrete.

Subtypes — any table-type in which the element data type of the subtype is the same as, or a subtype of, the element data type of the base table-type, the key data type of the subtype is the same as, or a subtype of, the key data type of the base table-type, and the limits on the subtype are such that base-minimum-size $\geq$ subtype-minimum-size, and subtype-maximum-size $\geq$ base-maximum-size.

Operations — Equal, Empty, IsPresent, Select, Insert, Delete.

Empty(): table(*key*) of *element* — yields the value which is the empty-set containing no pairs.

IsPresent(x: table (*key*) of *element*, y: *key*): Boolean — is true if there is a pair in the set x whose key member has value y, else fails.

Select(x: table (*key*) of *element*, y: *key*): element is — if IsPresent(x,y), then that value of the *element* data type which is the element member of the ordered pair in the set x whose key member has value y; else undefined.

Insert(x: table (*key*) of *element*, y: *key*, z: *element*): table (*key*) of *element* is — if Not(IsPresent(x, y)), then the set formed by inserting the pair (y, z) in the set x; else undefined.

Delete(x: table (*key*) of *element*, y: *key*): table (*key*) of *element* yields — if IsPresent(x, y), then the sequence formed by deleting from the set x the pair whose key member has value y, else the set x.

Equal(x, y: table (*key*) of *element*): Boolean is — true if for every value k of the *key* data type, Not(IsPresent(x, k)) implies Not(IsPresent(y, k)), and IsPresent(x, k) implies both IsPresent(y, k) and Select(x, k) = Select(y, k), else false.

NOTE 45 A Table-type is not just a List of Record(key, element), although its value space is a subset of the value space of that list-type. Table is characterized by the uniqueness constraint on occurrences of values of the key data 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 0607813 094

Annex E
(informative)
Bibliography

[1] **ISO/IEC CD 11404** — *Common Language-Independent Data Types.* Committee Draft.

[2] **CCITT430-520** — *CCITT Rec. T430 series and T520 series, Document transfer and manipulation (DTAM).*

[3] **IS07942:1985** — *Information processing systems, Computer graphics — Graphical kernel system (GKS) functional description.* International Standard.

[5] **IS08632:1987** — *Information processing systems, Computer graphics — Computer graphics metafile for the storage and transfer of picture description information (CGM).* International Standard.

[6] **IS09592:1989** — *Information processing systems, Computer Graphics — Programmers hierarchical interactive graphics system (PHIGS).* International Standard.

[7] **IS09636:1991** — *Information processing systems, Computer Graphics — Computer Graphics interfacing techniques for dialogues with graphical devices (CGI).* 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 0607814 T20

© ISO/IEC

ICS 35.140

Descriptors: data processing, information interchange, graphic data processing, image processing, communication procedure, specifications, operating requirements.

Price based on 69 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