

AS ISO/IEC 12087.3:2015

ISO/IEC 12087-3:1995

ISO/IEC 12087-3:1995/Amd 1:1996

AS ISO/IEC 12087.3:2015



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

Part 3: Image Interchange Facility (IIF)



This Australian Standard® was prepared by Committee IT-031, Computer Modelling and Simulation. It was approved on behalf of the Council of Standards Australia on 20 April 2015. This Standard was published on 12 May 2015.

The following are represented on Committee IT-031:

- ANZLIC—The Spatial Information Council
 - Department of Defence (Australia)
 - Simulation Australia
-

This Standard was issued in draft form for comment as DR AS ISO/IEC 12087.3:2015.

Standards Australia wishes to acknowledge the participation of the expert individuals that contributed to the development of this Standard through their representation on the Committee and through the public comment period.

Keeping Standards up-to-date

Australian Standards® are living documents that reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued.

Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments that may have been published since the Standard was published.

Detailed information about Australian Standards, drafts, amendments and new projects can be found by visiting www.standards.org.au

Standards Australia welcomes suggestions for improvements, and encourages readers to notify us immediately of any apparent inaccuracies or ambiguities. Contact us via email at mail@standards.org.au, or write to Standards Australia, GPO Box 476, Sydney, NSW 2001.

Australian Standard[®]

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

Part 3: Image Interchange Facility (IIF)

First published as AS ISO/IEC 12087.3:2015.

COPYRIGHT

© Standards Australia Limited

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher, unless otherwise permitted under the Copyright Act 1968.

Published by SAI Global Limited under licence from Standards Australia Limited, GPO Box 476, Sydney, NSW 2001, Australia

ISBN 978 1 76035 046 8

PREFACE

This Standard was prepared by the Standards Australia Committee IT-031, Computer Modelling and Simulation.

The objective of this Standard is to provide a data format specification known as the Image Interchange Facility (IIF) and an application program interface specification. This IIF data format may be used for image data interchange in open, heterogeneous environments. It may also serve as a local file format for imaging applications, especially in conjunction with AS ISO/IEC 12087-2.

This Standard is identical with, and has been reproduced from ISO/IEC 12087-3:1995, *Information technology—Computer graphics and image processing—Image Processing and Interchange (IPI)—Functional specification*, Part 3: *Image Interchange Facility (IIF)* and its Amendment 1 (1996) which has been added at the end of the source text.

As this Standard is reproduced from an International Standard, the following applies:

- (a) In the source text ‘this part of ISO/IEC 12087’ should read ‘this Australian Standard’.
- (b) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>	<i>Australian Standard</i>
ISO/IEC	AS ISO/IEC
12087 Information technology—Computer graphics and image processing—Image Processing and Interchange (IPI)—Functional specification	12087 Information technology—Computer graphics and image processing—Image Processing and Interchange (IPI)—Functional specification
12087-1 Part 1: Common architecture for imaging	12087.1 Part 1: Common architecture for imaging
12089 Information technology—Computer graphics and image processing—Encoding for the Image Interchange Facility (IIF)	12089 Information technology—Computer graphics and image processing—Encoding for the Image Interchange Facility (IIF)

Only normative references that have been adopted as Australian or Australian/New Zealand Standards have been listed.

The terms ‘normative’ and ‘informative’ have been used in this Standard to define the application of the annexes to which they apply. A ‘normative’ annex is an integral part of a Standard, whereas an ‘informative’ annex is only for information and guidance.

CONTENTS

1 Scope	1
2 Normative references	3
3 Definitions and abbreviations	5
3.1 Definitions	5
3.2 Abbreviations	5
4 The IPI-IIF architecture	6
4.1 The IPI-IIF Data Format and the IPI-IIF Gateway	6
4.2 Interworking between IPI-IIF Gateway and IPI-PIKS	7
5 The IIF data format (IIF-DF)	9
5.1 Basic features of the IIF-DF	9
5.1.1 Objects that are expressed in the IIF-DF	9
5.1.2 Syntax notation	9
5.1.3 Encoding of syntax entities	10
5.1.4 Rules that are not formally expressed within the IIF syntax	10
5.2 Structure of the IIF-DF syntax	11
5.2.1 Overall structure	11
5.2.2 Image structures	12
5.2.3 Placement of pixel fields	14
5.2.4 Encoding of pixel fields	14
5.2.5 Attributes, annotations, and image-related data	14
5.3 Syntax entities of the IIF-DF	16
5.3.1 Entities for the description of the entire IIF-DF	19
5.3.2 Entities for the description of images	27
5.3.3 Entities for the description of the representation of pixel values	46
5.3.4 Entities for the description of image-related data	59
5.3.5 Entities for the description of image attributes	79
5.3.6 Entities for the description of image annotations	111
5.3.7 Entities for the description of basic data objects	114

	<i>Page</i>
6 IPI-IIF Conformance	121
6.1 Standardized profiles for the IIF-DF	121
6.1.1 Full PIKS profile of the IIF-DF	122
6.1.2 Foundation profile of the IIF-DF	125
6.2 Registered profiles for the IIF-DF	128
6.2.1 Application-specific semantics	128
6.2.2 Constraining methods	129
6.3 Extension methods	130
7 IPI-IIF Gateway functionality	131
7.1 Basic categories of IPI-IIF Gateway functions	131
7.1.1 Gateway control and error handling	131
7.1.2 Import and export functionality	132
7.1.3 Parse and generate functionality	132
7.1.4 Data structure access functionality	133
7.1.5 Data structure manipulation functionality	134
7.1.6 Compression and decompression functionality	134
7.1.7 Application-oriented functionality	137
7.2 IPI-IIF gateway-internal tables	137
7.3 Survey of IPI-IIF Gateway functions	138
7.4 IPI-IIF Gateway functionality by manual pages	139
7.5 PIKS-IIF interworking protocol	193
 Annexes	
A List of IIF-DF syntax entities and component names (normative)	194
B List of IPI-IIF Gateway function-caused errors (informative)	208
C Typical IIF image interchange scenario (informative)	210
D Examples of IIF-DF images (informative)	212
D.1 Simple binary image	213
D.2 Colour image with colourimetric attributes	214
D.3 Tiled image	216
E Example program for the use of the IPI-IIF Gateway (informative)	218
F IIF-DF syntax diagrams (informative)	221
G Bibliography	243

INTRODUCTION

ISO/IEC 12087-1 establishes the conceptual and architectural framework for ISO/IEC 12087. In particular, it defines the types of all image data objects, image-related data objects, and attributes that may be interchanged by means of the IPI-IIF.

ISO/IEC 12087-2 establishes the specification of the Programmer's Imaging Kernel System (IPI-PIKS).

ISO/IEC 12087-3 provides a data format specification and an application program interface specification. The IIF data format may be used for image data interchange in open, heterogeneous environments. It may also serve as a local file format for imaging applications, especially in conjunction with ISO/IEC 12087-2. In future, the IIF data format could be used by telecommunication standards. Examples are future versions of File Transfer, Access, and Management (FTAM), ISO/IEC 8571; the Message Oriented Text Interchange Systems (MOTIS), ISO/IEC 10021 (also known as Message Handling System (MHS), CCITT Recommendation X.400). Thus the IIF data format could become part of application-oriented OSI communications protocols.

Within the IIF data format (IIF-DF), compressed images may be specified and interchanged. For this purpose, the following standards are referenced:

- CCITT Recs. T.4 and T.6 (Facsimile)
- ISO/IEC 11544 (JBIG)
- ISO/IEC 10918 (JPEG)
- ISO/IEC 11172 (MPEG-1)

Image data streams that conform to the encoded representation of compressed image data specified by these standards may be included in the IIF-DF. For instance, a time series image can be represented as an array of time slices, each of which is encoded according to the JPEG Standard. Furthermore, the IIF-DF allows images to be represented through the combination of compressed parts with uncompressed parts. It is also possible to use multiple compression methods within a single IIF-DF-conformant image. For instance, a colour image can be represented as tiled images whereby some tiles are encoded according to the lossy mode of the JPEG Standard and others according to the lossless mode. For detailed information concerning compressed data streams and compression/decompression functionality, refer to 5.3.3 and 7.1.6, respectively.

There are various possibilities for interaction and data exchange between the IPI-PIKS domain and the IPI-IIF domain. Both domains are controlled by the application via application program interfaces (APIs). For a detailed description of the interworking between the IPI-PIKS and the IPI-IIF refer to clause 4 (the IPI-IIF architecture) and clause 7 (the IPI-IIF Gateway functionality). For a description of the relation between the types of objects that may be interchanged by means of the IPI-IIF and those types of objects that may be processed by the IPI-PIKS, refer to clause 6 (the profiles for the IIF data format). Refer also to ISO/IEC 12087-1.

NOTES

AUSTRALIAN STANDARD

Information technology—Computer graphics and image processing—Image Processing and Interchange (IPI)—Functional specification**Part 3:
Image Interchange Facility (IIF)****1 Scope**

This part of ISO/IEC 12087 facilitates the interchange of digital images. For this purpose, conceptual, architectural, and functional definitions of the Image Interchange Facility (IPI-IIF) are established. ISO/IEC 12087-3 consists of two major parts, the:

- a) IIF data format (IIF-DF) definition (by means of a formal syntax, described according to the Abstract Syntax Notation One (ASN.1) -- refer to clause 5), and the
- b) IIF Gateway definition (by means of a manual page description of the functionality of an Application Program Interface (API) -- refer to clause 7).

An IPI-IIF-conformant implementation has to fulfill the functionality specification of the IIF Gateway, as outlined in clause 7. Besides the IIF Gateway, there may be information processing systems (software such as parsers, generators, etc.) which read and/or write the IIF-DF.

The IPI-IIF is based on the definitions described in ISO/IEC 12087-1, the "Common Architecture for Imaging". The IPI-IIF, as a whole, may be characterized briefly as follows:

- c) By means of the IIF data format and Gateway, image data objects and image-related data objects are transported to and from application environments.
- d) By means of the full PIKS profile of the IPI-IIF data format (i.e., a format for data interchange between IPI-IIF and IPI-PIKS), image data objects and image-related data objects are imported to and exported from the Programmer's Imaging Kernel System (IPI-PIKS), defined in ISO/IEC 12087-2.
- e) The IPI-IIF facilitates the storage of image data objects and image-related data objects in a variety of pre-defined storage modalities, including different periodicity organizations, such as pixel-interleaving or band-interleaving.
- f) This part of ISO/IEC 12087 defines syntax of image data (and image-related data) streams. The encoding of IIF data types is defined in ISO/IEC 12089. See also 5.3.3.
- g) The IPI-IIF supports a concept of standardized conformance profiles. Initially, three conformance profiles are defined within ISO/IEC 12087.
- h) An IIF data stream may be stored in devices such as file systems. An IIF data stream may be interchanged and communicated in data networks (e.g., LANs and WANs) or in other data communication facilities. All low-level data storage and transfer is delegated, for instance, to the operating system of the target hardware.
- i) The IIF Gateway performs compression and decompression of image data objects using standardized compression and decompression techniques. These techniques are referenced in this part of ISO/IEC 12087. See 1.4.5 and 5.3.3 and 7.5 for further definition.

- j) The IIF Gateway is accessible via an API to perform image interchange functions. See clause 7 for a definition of IIF Gateway functionality.

Reference shall be made to this part of ISO/IEC 12087, and its definitions shall be employed, whenever images are interchanged, according to the IPI-IIF, among different imaging applications environments or among imaging devices. The IPI-IIF is applicable to scenarios requiring the interchange of digital images, as outlined in Annex C.

The use of the IIF data format as a superset of the functionality of most of the existing image interchange formats solves the problem of application-independent syntactical and semantical interpretation and understanding of image data.

The IPI-IIF is applicable to image interchange in and among different application domains. The following application areas have been considered:

- Medical imaging
- Remote sensing
- Publishing
- Industrial vision
- Computer graphics arts
- Computer animation
- Scientific visualization
- Mission planning
- Document processing
- Outdoor scene surveillance

The limiting of the IPI-IIF scope to certain application domains is a matter of profiling. This is treated in clause 6.

NOTE - Whether an image interchange format may also be regarded as a device format, depends on the (local) processing power of the device itself. Thus a conceptually "high-level" format which has become an industrial standard page description language for desktop electronic publishing, can be regarded as a device format. The IPI-IIF may well be considered a device format if, for instance, there is an IPI-IIF-compatible printer which is able to receive, process, and hardcopy an image according to the IPI-IIF. In the same sense, it is reasonable to design IPI-IIF-compatible image sources, e.g. IPI-IIF camera systems.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 12087. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 12087 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 2022:1986, *Information processing - ISO 7-bit and 8-bit coded character sets - Code extension techniques*.

ISO/IEC 8613:1994, *Information processing systems - Text and office systems - Open Document Architecture (ODA) and Interchange Format (ODIF)*.

ISO/IEC 8632:1992, *Information processing systems - Computer graphics - Metafile for the storage and transfer of picture description information*.

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

ISO/IEC 9069:1988, *Information processing systems - SGML support facilities - SGML Document Interchange Format (SDIF)*.

ISO/IEC TR 10000-1:1990, *Information technology - Framework and taxonomy of International Standardized Profiles - Part 1: Framework*.

ISO/IEC TR 10000-2:1994, *Information technology - Framework and taxonomy of International Standardized Profiles - Part 2: Principles and taxonomy for OSI Profiles*.

ISO/IEC 10031-1:1991, *Information technology - Text and office systems - Distributed office application model - Part 1: General model*.

ISO/IEC 10031-2:1991, *Information technology - Text and office systems - Distributed office application model - Part 2: Distinguished object reference and associated procedures*.

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

ISO/IEC 10918-2: To be published., *Information technology - Digital compression and coding of continuous-tone still images - Part 2: Compliance testing*.

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.*

ISO/IEC 11172-2:1993, *Information technology - Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s - Part 2: Video.*

ISO/IEC 11172-3:1993, *Information technology - Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s - Part 3: Audio.*

ISO/IEC 11544:1993, *Information technology - Coded representation of picture and audio information - Progressive bi-level image compression.*

ISO/IEC 12087-1:1995, *Information technology - Computer graphics and image processing - Image Processing and Interchange (IPI) - Functional specification - Part 1: Common architecture for imaging.*

ISO/IEC 12089:—¹⁾, *Information technology - Computer graphics and image processing - Encoding for the Image Processing and Interchange Standard (IPI) - Encoding for the Image Interchange Facility (IIF).*

CCITT Rec. G.711(1984), *Coding of analogue signals by pulse code modulation.*

CCITT Rec. G.721(1984), *32 Kbit/s Adaptive Differential Pulse Code Modulation (ADPCM).*

CCITT Rec. T.4(1988), *Standardization of Group 3 Facsimile Apparatus for Document Transmission.*

CCITT Rec. T.6(1988), *Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus.*

CCITT Rec. T.30(1988), *Procedures for Document Facsimile Transmission in the General Switched Telephone Network.*

NOTES

1 All normative references which are common to Parts 1 to 3 of ISO/IEC 12087 are included in ISO/IEC 12087-1. In ISO/IEC 12087-3, only the IIF-specific references are listed.

2 References to documents which are neither ISO/IEC Standards nor CCITT Recommendations are given in Annex G.

3 Some ISO Standards are technically aligned with CCITT Recommendations, in particular the ASN.1 Standard (ISO Standards 8824/8825 and CCITT Recs. X.208/X.209). The differences between the International Standard definitions and the CCITT definitions are quite small, and should not affect interoperability between implementations written against either document. Within this part of ISO/IEC 12087, the ISO Standards are referenced whenever possible.

1) To be published.

3 Definitions and Abbreviations

3.1. Definitions

For the purpose of this part of ISO/IEC 12087, the definitions given in ISO/IEC 12087-1 apply.

3.2. Abbreviations

API	Application Program Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CGM	Computer Graphics Metafile
DCT	Discrete Cosine Transform
DOAM	Distributed Office Application Model
DOR	Distinguished Object Reference
FOD	Interchange Format and Representation Profile of Office Documents
FTAM	File Transfer, Access and Management
IIF-DF	Image Interchange Facility - Data Format
IPI	Image Processing and Interchange
IPI-CAI	IPI - Common Architecture for Imaging
IPI-IIF	IPI - Image Interchange Facility
IPI-PIKS	IPI - Programmer's Imaging Kernel System
JBIG	Joint Bi-level Image Experts Group
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
MHS	Message Handling System
MOTIS	Message-Oriented Text Interchange Systems
MPEG	Moving Pictures Experts Group
ODA	Open Document Architecture
OSI	Open Systems Interconnection
PER	Packed Encoding Rules
SDIF	SGML Document Interchange Format
SGML	Standard Generalized Markup Language
WAN	Wide Area Network

4 The IPI-IIF Architecture

4.1 The IPI-IIF Data Format and the IPI-IIF Gateway

As outlined in clause 1, the IPI-IIF consists of the specification of a data format and the specification of functionality. The data format is called Image Interchange Facility Data Format (IIF-DF). It is described in clause 5. Clause 6 describes conformance profiles for the IIF-DF.

The functional component of the IPI-IIF is called IPI-IIF Gateway. It is described in clause 7. The IPI-IIF Gateway functions are called by an application program via a specific API (Application Program Interface). Concerning data interchange, it provides functionality for

- a) the import of image data to and export of image data from application, as well as
- b) the import of image data to and export of image data from the IPI-PIKS.

Part of the IPI-IIF Gateway functionality deals with the differing complexity of the IPI-IIF data types (clause 6 of ISO/IEC 12087-1) and the IPI-PIKS data types (clause 5 of ISO/IEC 12087-1). The IPI-IIF data types are defined according to the (generic) IPI data types that are introduced in clause 4 and form a superset of the IPI-PIKS data types. The IPI-IIF data types support compound and heterogeneous image structures of arbitrary hierarchical organization whereas the IPI-PIKS data types support five-dimensional images with limited heterogeneity. The IPI-IIF Gateway provides a general mechanism - called attach/detach functionality - to combine simpler IIF structures into more complex ones, and extract simpler structures from more complex ones. Hence, this functionality represents the interface between the IPI-IIF data types (that are interchangeable by means of the IPI-IIF) and the IPI-PIKS data types (that can be processed by the IPI-PIKS, but also interchanged by the IPI-IIF).

The IPI-IIF Gateway function classes are:

- c) Gateway control functionality: These functions are used to open and close the IPI-IIF Gateway, to inquire about its status, and to handle errors.
- d) Import and export functionality: These functions allow for the import and export of image and image-related data to and from the IPI-IIF Gateway via the application program.
- e) Parse and generate functionality: These functions allow for the translation between IPI-IIF data streams (according to the IPI-IIF data format) and IPI-IIF gateway-internal data structures that are accessible and manipulable via IPI-IIF Gateway functions of category f) and g), respectively.
- f) Data structure access functionality: These functions allow for the access of parsed image data structures. This includes tree traverse, inquiry, put value, and get value functions.
- g) Data structure manipulation functionality: These functions allow for the manipulation of image structures. Create, delete, attach, and detach functions are provided.
- h) Compression and decompression functionality: Functions are provided for the compression and decompression of pixel fields according to the standards listed in section 1.4.3.
- i) Application-oriented functionality: This category encompasses functions which perform at the same time multiple functional steps according to categories c) to h). Seen from the application, these functions are located on a higher (and thus more application-oriented) level.

The specification of the functions is given in clause 7.

4.2 Interworking between IPI-IIF Gateway and IPI-PIKS

The interworking (i.e. data flow and function calls) of the IIF Gateway and the IPI-PIKS for image data interchange is depicted in Figure 1. The diagram shows three domains, called the "application domain," the "IIF Gateway domain," and the "IPI-PIKS domain." These "domains" indicate whether a certain function or data structure is part of the IPI-PIKS, the IIF Gateway, or the application program, respectively. The arrows between (squeezed) ovals indicate the data flow. Possible interworking situations include:

- a) The imaging application program uses only an IIF Gateway, but no IPI-PIKS: The application program may import and export image data which is interchanged using the IIF data stream. On the other hand, the application program has the capability to process image data by means of application-private imaging functions.
- b) The imaging application program uses a non-IIF-capable IPI-PIKS and no IIF Gateway: In this case, the IPI-PIKS conformance profiles defined in ISO/IEC 12087-2 describe the import and export functionality of the IPI-PIKS.
- c) The imaging application program uses an IIF-capable IPI-PIKS and no IIF Gateway: In this case, the standardized import and export functions of the IPI-PIKS, if called by the application program, produce image data according to the full PIKS profile or the foundation profile of the IPI-IIF data format. The application program will be able to read data streams that conform to the profile of the IPI-IIF data format. The application program, as well as the import and export function of the IPI-PIKS implementation, will have a parser and a generator for the IPI-IIF data format.
- d) The imaging application program uses both an IPI-IIF Gateway and an IPI-PIKS: The import/export, parse/generate and data structure access and manipulation functions work according to a) and b), above. Further details are provided by the manual pages for these function descriptions (see clause 7).

Besides, it is possible that the application program does not use the standardized import and export functions, but explicitly ("manually") controls the interworking of the IPI-IIF Gateway and the IPI-PIKS by using only the import and export functions of the IPI-IIF Gateway and the IPI-PIKS utility functions. In this case, image data is transferred by the `get_pixel` and `put_pixel` functions from the IPI-PIKS to the application domain, and vice versa.

If both the IPI-IIF Gateway and the IPI-PIKS are developed and supplied as one software unit, the IPI-IIF Gateway and IPI-PIKS APIs may be linked together. In this case, the internal interworking between the IPI-IIF Gateway and the IPI-PIKS may be designed to be more efficient. In either case, private, non-standardized image data input and output may take place as shown by the dashed arrows in Figure 1. It is up to the application program to manage the reading and writing of private image file formats and convert the data into the IPI-IIF data format. See also 7.1.2 for the specification of low-level data exchange mechanisms of the IPI-IIF Gateway.

The IPI-PIKS functions relevant for the interworking of the IPI-PIKS with the IPI-IIF Gateway (defined in ISO/IEC 12087-2) are:

- e) `input_object`: data transfer from the IPI-IIF Gateway to the IPI-PIKS.
- f) `output_object`: data transfer from the IPI-PIKS to the IPI-IIF Gateway.

The IPI-PIKS can write and read image data streams only by means of the above-mentioned `input_object/output_object` functions. However, an application program may feed data to and from the IPI-PIKS with other utility functions, such as `get_pixel`, `put_pixel`, and `import/export`. For further details, see ISO/IEC 12087-2.

The functionality of both the IPI-IIF Gateway and the IPI-PIKS, as indicated above, is accessible via two separate APIs from the application program.

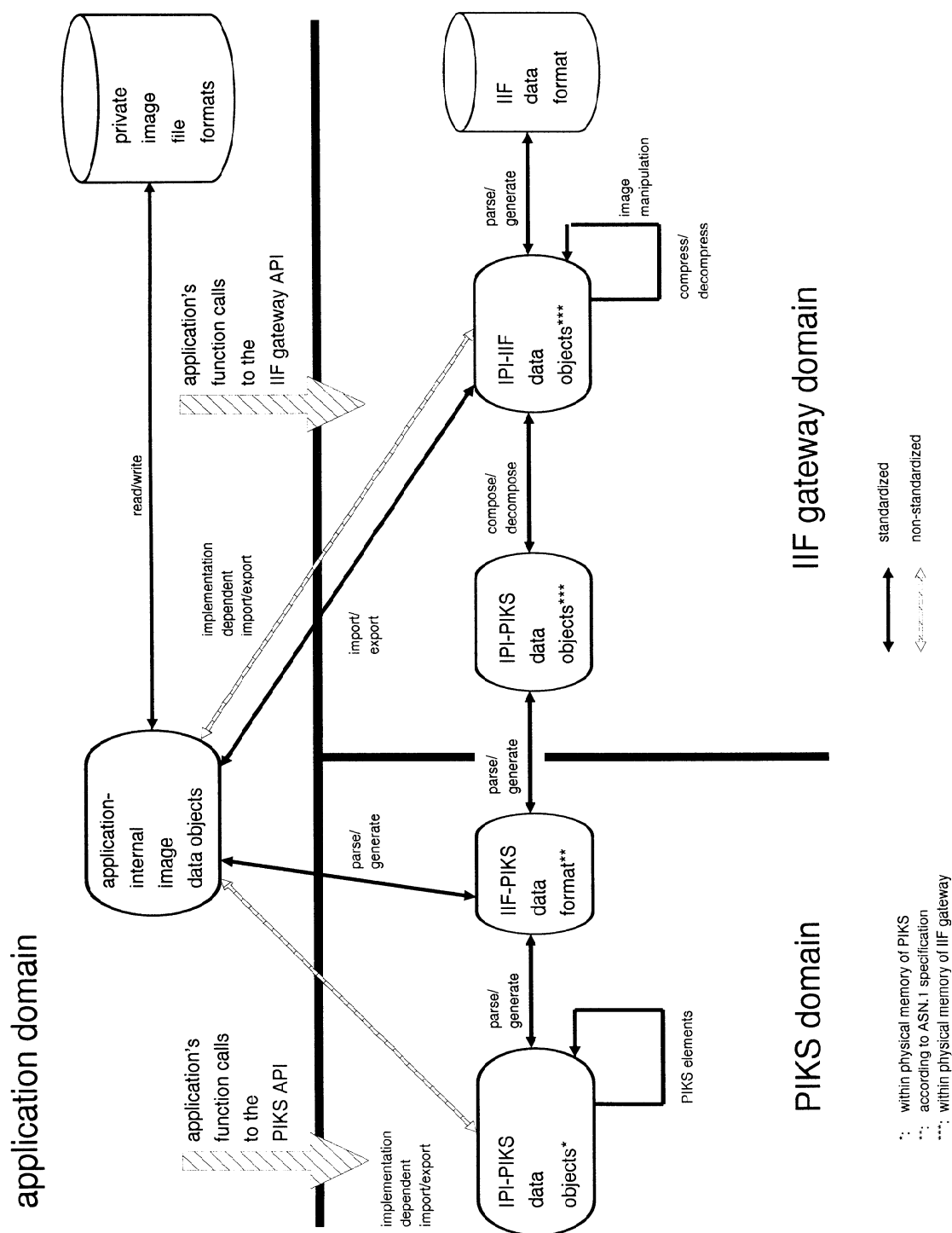


Figure 1 - Interworking of the IPI-IIF Gateway and the IPI-PIKS for image data interchange

5 The IIF data format (IIF-DF)

5.1 Basic features of the IIF-DF

This Clause gives an overview of:

- the kind of data objects which may be expressed by the IIF-DF,
- the kind of formal description method which has been chosen, and
- which encoding rules shall apply, in order to produce IIF-DF-conformant data streams.

Clause 5.3 defines the full IIF-DF syntax by giving the production rules for all data entities. Within this part of ISO/IEC 12087, the rules of the IIF-DF syntax are arranged in preorder traversal (depth-first) with the exception of image attributes, which are described after the non-image data types.

5.1.1 Objects that are expressed in the IIF-DF

The IIF-DF syntax allows description of a sequential format for all data objects that are structured according to the IPI-IIF data types, as defined in ISO/IEC 12087-1, clause 6. Since the IPI-PIKS data types (ISO/IEC 12087-1, clause 5) form a proper subset of the IPI-IIF data types, they are covered by the IIF-DF too. The names of the syntax entities correspond to the names of the data types of ISO/IEC 12087-1. The structure and semantics of these data entities are defined with reference to ISO/IEC 12087-1.

NOTE - ISO/IEC 12087-1 provides IPI-IIF data types and IPI-PIKS data types. Both are derived from the (generic) IPI data types. The IPI-IIF data types represent a very broad approach to image data modelling. The IPI-PIKS data types are oriented by the needs of image processing as defined in ISO/IEC 12087-1.

In addition to the data types defined in ISO/IEC 12087-1, the IIF-DF contains some data entities that specify the placement of pixel data, the ordering of elements of multi-dimensional arrays, and the coded representation of elementary data types, including compression algorithms.

5.1.2 Syntax notation

The syntax is expressed in ASN.1 (Abstract Syntax Notation One), according to ISO/IEC Standard 8824, "Specification of Abstract Syntax Notation One (ASN.1)." ASN.1 is a formal description language. It defines a set of primitive data types, such as INTEGER, ENUMERATED, and REAL and provides a facility to construct new elements with their own typing inherent in the structure using the constructors SEQUENCE, SEQUENCE OF and CHOICE. This allows for new data types to be defined which are uniquely recognizable within an application. To make these definitions more readable, textual labels may be associated with the elements in a constructor type. In order to distinguish different occurrences of the same type within one constructor, various types of tags are provided that may be associated with the constructor's elements.

Within the semantical description of the IPI-IIF data format, each element (which is either a primitive data type or a constructed type) is called *syntax entity*. According to ASN.1, the names of the syntax entities begin with capital letters. An example for a syntax entity is *BandRecord*. Syntax entities consist of a number of *components*. According to ASN.1, the component labels begin with lower case letters. For example the *BandRecord* entity consists of the *number-of-bands* component, the *data-placement* component, and the *record-components* component. The type of each component is defined by reference to another syntax entity which is either described elsewhere in the IIF-DF syntax or within the ASN.1 standard. For example, the *data-placement* component is represented by the *DataPlacement* entity, which is described within the IIF-DF syntax. The *number-of-bands* component is represented by the *INTEGER* entity, which is an elementary syntax entity. For its definition, refer to ISO/IEC 8824. From the viewpoint

of the *BandRecord* entity, the *DataPlacement* entity is called *subentity*.

NOTE - Only those constructed ASN.1 data types for which a sequential order is defined are used within ISO/IEC 12087. Thus the IIF syntax always describes an unambiguously ordered data stream. However, concerning the placement and periodicity organization of pixel fields, the IIF-DF provides a maximum degree of flexibility.

5.1.3 Encoding of syntax entities

The encoding of IIF syntax entities is defined in ISO/IEC 12089.

NOTES

1 The encoding of IIF syntax entities as defined in ISO/IEC 12089 employs the Basic Encoding Rules for ASN.1 (BER), that are defined in ISO/IEC 8825. The encoding of a large pixel data field according to the BER can produce considerable space and processing time overhead when every pixel is expressed as an elementary ASN.1 data entity, consisting of a "tag" and a "length" field that precedes the "value" field. For this reason, ISO/IEC 12089 provides some methods on how to encode fields of pixel values as single entities in terms of ASN.1. This refers to the *ExternallyDefinedDataUnit* entity.

2 The BER provide a definite form and an indefinite form for the encoding of sequences of entities. Both are indicated by the keyword "SEQUENCE OF." In the definite form the number of content octets is given at the beginning of the sequence. In the indefinite form the number of content octets shall be calculated by parsing the whole sequence until the end-of-sequence token. Within the IIF-DF syntax, for some "SEQUENCE OF" constructs, a "number-of-..." component is provided. It specifies the number of entities (e.g., pixel values) that are present in the following sequence. This allows implementations to make use of the number of entities in advance (e.g., for memory allocation), regardless of what encoding method has been chosen.

5.1.4 Rules that are not formally expressed within the IIF syntax

There are rules and constraints within 5.3 that are needed in addition to the IIF syntax to define a proper IIF-DF. These rules and constraints are given by an accompanying text. They need to be checked within the parse and generate functions of the IPI-IIF Gateway, as illustrated in Figure 2.

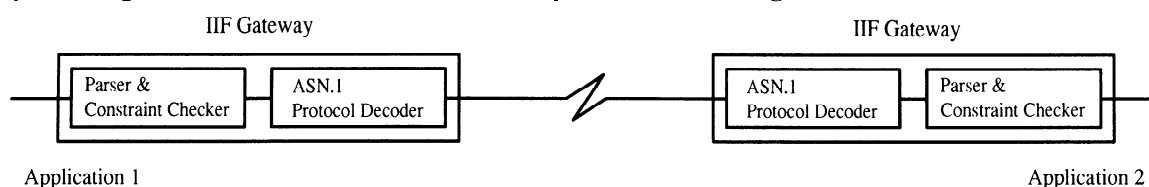


Figure 2 - IIF-DF parser and constraint checker within the IPI-IIF Gateway

NOTE - The IIF syntax is expressed using formal ASN.1 notation, but not all rules or constraints that determine the desired "correctness" of an IIF-DF-conformant data stream can be expressed in terms of this syntax. For example, there is no way to define a syntax that prevents the production of pixel arrays that contain more or fewer pixels than defined in the array's type definition. This deficiency, which is a result of the context-free grammar, is only one example. Other examples of constraints that need to be defined "on top" of the IIF syntax result from the desire for maximum flexibility in pixel data placement. In general, the number of constraints that need to be defined in addition to the syntax is a result of the tradeoff between flexibility and formal elegance. Furthermore, one may distinguish between various levels of constraints. Those constraints that determine, for instance, the completeness of pixel fields, are of a different nature as compared to constraints that deal with inconsistent colour attributes.

5.2 Structure of the IIF-DF syntax

This clause gives an overview of the structure of the IIF-DF syntax. The reader may study the following description of major IIF-DF syntax entities in conjunction with the three example images provided in Annex D and the IIF-DF syntax diagrams provided in Annex F.

5.2.1 Overall structure

The *FullDataFormat* entity is the top-most entity of the IIF-DF syntax tree. All other entities branch from there. There are two types of information contained in the *FullDataFormat* entity. First, there is information identifying the data as conforming to a particular version, conformance level, and application profile of the IIF. This information is contained in the *FormatDescriptor*, *Version*, and *Profile* entities. There is also a place to store some application-specific header information in the *ContentsHeader* entity. Second, there is the data content.

NOTE - Compound syntax entities may consist of an arbitrarily deep tree of sub-entities. Thus the term "something is contained (or stored) in a given syntax entity" does not imply that the information is contained in the named entity directly. It rather means that it is contained in either the entity or one of its subentities.

The data stored in the *ContentsBody* element can be images, image-related data, attributes, annotations, or basic data objects, as defined in ISO/IEC 12087-1. The overall structure of the IIF-DF syntax is shown in Figure 3.

NOTE - Figure 3 does not represent a complete syntax diagram. It is an abstract view of the IIF-DF syntax. Many details are left out. The solid boxes and arrows represent one possible instance of a syntax tree. The dashed boxes and arrows indicate potential alternatives according to the choices provided by the respective syntax entities. For an exact description of the IIF-DF syntax, refer to 5.3.

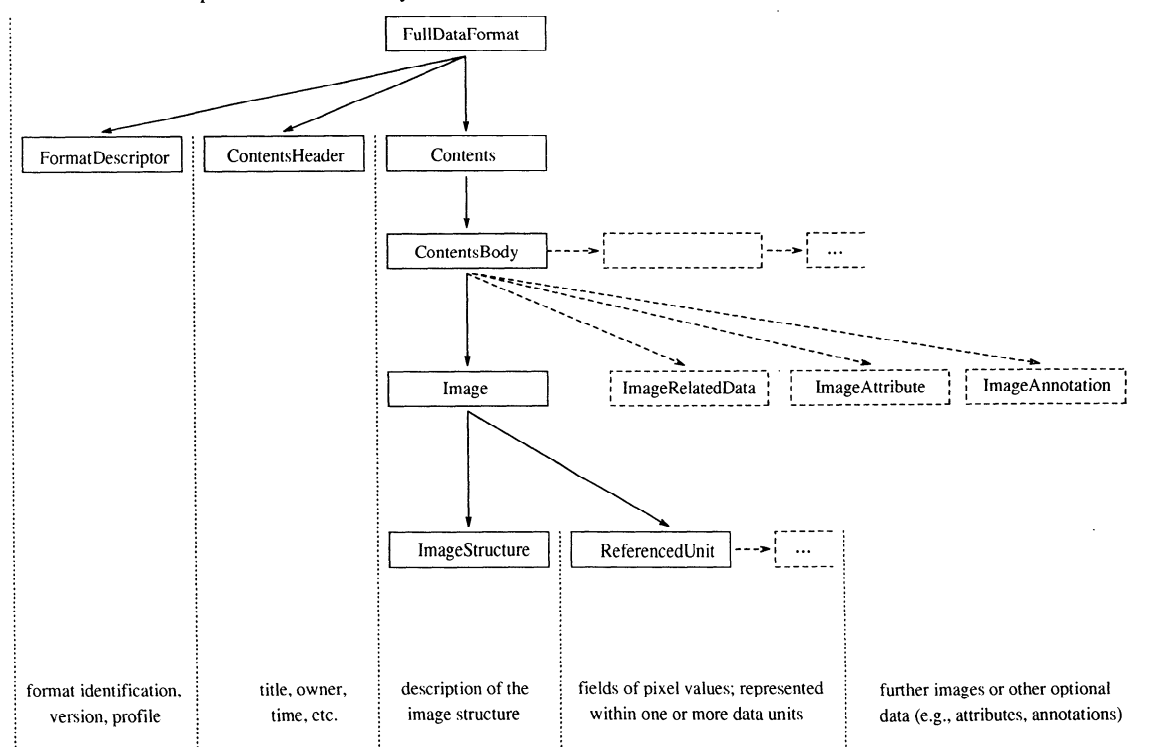


Figure 3 - Overall structure of the IIF-DF syntax.

5.2.2 Image structures

Images are stored in the *Image* entity and are split into structural information and data, called *image-structure* and *image-data*, respectively. As specified in the *ImageStructure* entity, an image can be either a compound image (such as arrays, records, lists or sets of other images) or a fundamental image. A fundamental image is an n-dimensional array of pixels which may consist of one or more bands. Images can have attributes and annotations associated with them.

The *CompoundImageArray* entity is used to specify an n-dimensional array of images; component images are accessed via an n-dimensional index. All the images stored in an array must have identical structures (including attributes, annotations, and image related data). The *CompoundImageRecord* entity is used to specify a record of images. The components are accessed via associated names, called *component-identifiers*. In contrast to arrays of images, the components are not required to have identical image structures. The *CompoundImageList* entity is used to specify a list of images of identical structure which are accessed via their position in the list. The *CompoundImageSet* entity is used to specify a set of images of identical structure. The syntax of the *ImageStructure* entity is recursive, i.e., the array elements, record components etc. are again of *ImageStructure* type.

The *FundamentalImageStructure* entity specifies the structure and sequential organization of an n-dimensional multiband array of pixels, along with associated image-related data, such as look-up tables, histograms, etc. It and its sub-entities provide enough flexibility to specify a wide range of pixel periodicity arrangements (e.g. pixel interleaved, band interleaved, tiled images, etc.). This is specified primarily by the *BandRecord* and *MetricArray* entities as described below.

The *BandRecord* entity is used to specify pixel arrays on a band-by-band basis; this entity is used to specify a band-interleaved image. The *MetricArray* entity is used to specify arrays of pixels or tiles. Pixels can be represented as a collection of bands in the *PixelStructure* entity. Thus, pixel-interleaved images are specified using a *MetricArray* of *PixelStructures*. The syntax of the *MetricArray* and *BandRecord* entities allow for arbitrary recursions. Thus, tiled images may be specified by constructing an array of arrays using the *MetricArray* entity for both hierarchical levels. Band-interleaved tiles are specified by a *MetricArray* containing *BandRecords*. Pixel-interleaved tiles are specified by a *MetricArray* containing *MetricArrays*. The recursive organization within the *FundamentalImageStructure* entity is illustrated in Annex F.

Entities specifying the structure of arrays in the IIF, such as *MetricArrays* and *CompoundImageArrays*, contain components specifying the number of dimensions of the array, the size of the array along each dimension, and a unique identifier for every dimension. The *Serialization* entity which is also part of every array description, indicates which axis varies fastest, second fastest, and so on within the sequence of pixel values.

The various hierarchical levels of an image structure described by the IIF-DF syntax are shown in Figure 4.

NOTE - Figure 4 does not represent a complete syntax diagram. It is an abstract view of the IIF-DF syntax. Many details are left out. The solid boxes and arrows represent one possible instance of a syntax tree. The dashed boxes and arrows indicate potential alternatives according to the choices provided by the respective syntax entities. For an exact description of the IIF-DF syntax, refer to 5.3.

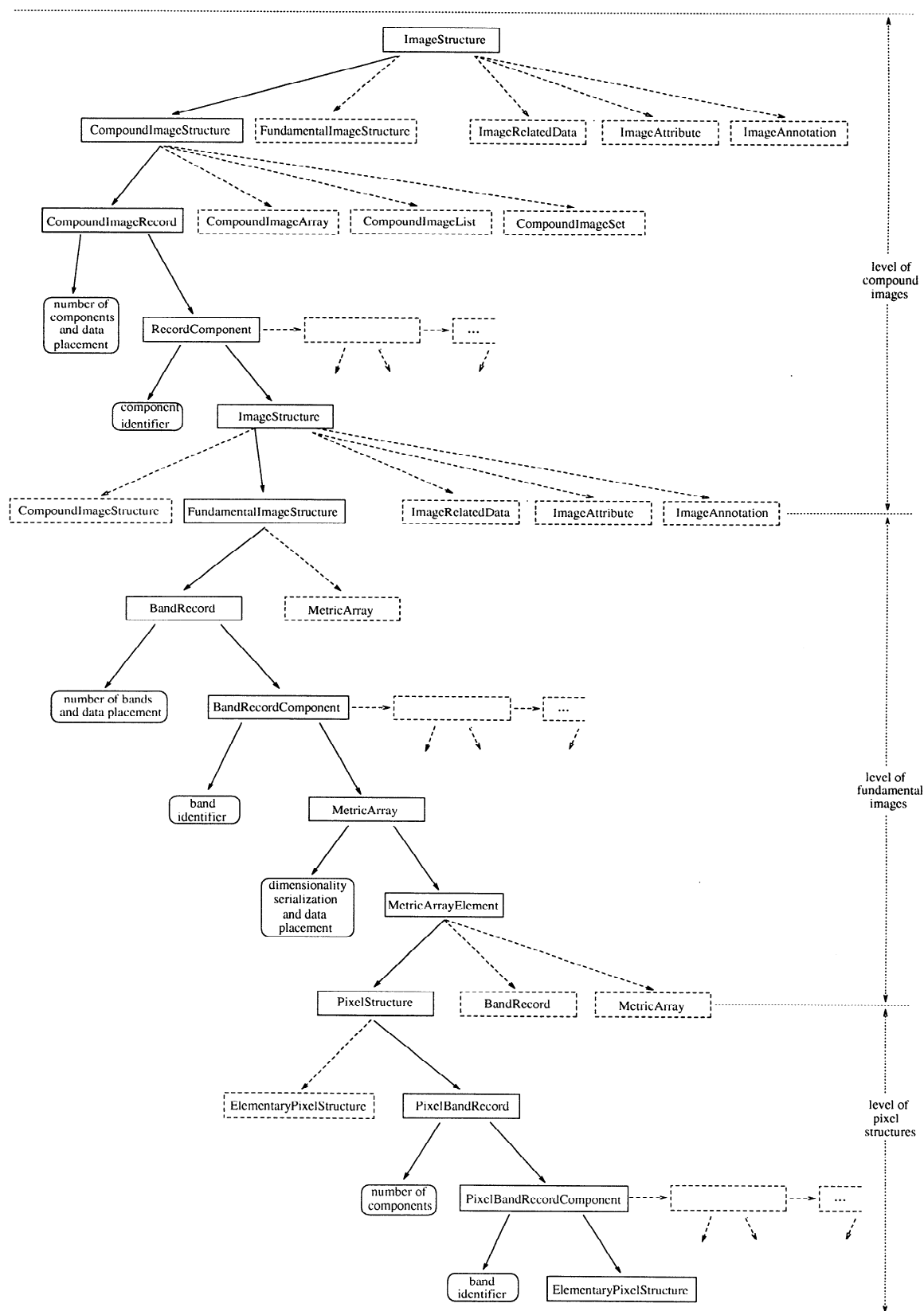


Figure 4 - Representation of image structures within the IIF-DF syntax.

5.2.3 Placement of pixel fields

As described above, the *image-structure* component of the *Image* entity and its subentities only contain information describing the image structure. The associated fields of pixel values are stored in the *image-data* component. Various modes of data placement and various pixel field encodings are permitted. The "atomic portion" of pixel data (a set of data that is always stored as one unit) is called a *ReferencedUnit* (see Figure 3).

The *DataPlacement* entity associated with each of the image structure entities is used to indicate whether the pixel values associated with the image structure are stored together in a single entity, called *ReferencedUnit*, or whether they are distributed into multiple entities. There are three cases that must be delineated.

- a) The data associated with the structure is divided into multiple *ReferencedUnit* entities.
- b) The data associated with the structure is completely contained in a single *ReferencedUnit* entity; additionally this entity does not contain any data from other structures.
- c) The data associated with this structure is contained in a single *ReferencedUnit* entity along with data from other structures.

In either case, the pixel fields of an image are placed at an arbitrary position behind the image structure tree. All of these data sets are modelled in the syntax as a *SEQUENCE OF ReferencedUnit*.

Each *ReferencedUnit* starts with a label that indicates what portion of the pixel values of the image it contains. In principle, the label specifies the path from the root of the image structure tree down to the subentity which describes the structure of the pixel data contained in the referenced unit. For a definition of the *ReferencedUnit* entity and the construction of labels refer to 5.3.3.

NOTE - The labelling of referenced units and all other labellings within the IIF-DF as well are based on semantical identifiers. Using ASN.1 as syntax notation, the syntactical definition of the IIF-DF becomes independent of encoding rules. Thus, mechanisms such as byte counts or byte offsets are excluded.

5.2.4 Encoding of pixel fields

For the encoded representation of pixel fields, four choices are provided within the *SingleDataUnit* entity:

- 1) The *BuiltinEncodedDataUnit* entity uses elementary ASN.1 syntax entities, such as INTEGER and REAL to encode every pixel value.
- 2) The *ExternallyDefinedDataUnit* entity uses techniques defined in ISO/IEC 12089 to store the pixels in a single ASN.1 OCTET STRING.
- 3) The *CompressedDataUnit* entity allows standard compression techniques, such as JBIG and JPEG to be used to encode the pixel information.
- 4) The *RegisteredDataUnit* entity permits the transportation of registered encodings through the use of an ASN.1 ANY type.

Refer to 5.3.3 for detailed information on each of the above-mentioned encoding alternatives.

5.2.5 Attributes, annotations, and image-related data

The IIF-DF supports the representation of all image attributes and image-related data types that are defined in ISO/IEC 12087-1.

ImageAttribute entities can be attached to image structures or can be represented independently.

MetricDescription entities are used to specify the size and type of coordinate systems from which the image was sampled. *ChannelCharacteristics* entities are used to specify the transfer characteristics, precision, and semantics of the pixel values. The *ColourRepresentation* entity is used to specify colour spaces, both standard and non-standard. Finally, a freeform attribute is provided to allow application specific attributes to be represented.

Within the *MetricDescription* entities identifiers are used to refer to specific image dimensions which are defined within the image structure declaration. Within the *ChannelCharacteristics* and *ColourRepresentation* entities, the same identifier-based mechanism is used to refer to specific image bands. The former identifiers are called *dimension-identifiers*, the latter *band-identifiers*.

ImageRelatedData entities can also be attached to images or represented independently. Types of image related data include look up tables, histograms, regions of interest, neighbourhoods, static arrays, feature lists, and value bounds collections. These data types are used by ISO/IEC 12087-2.

ImageAnnotation entities can also be attached to images or represented independently. Types of pre-registered image annotations include: plain text, structured text according to SGML and ODA, graphics contents according to CGM, and audio data streams according to CCITT Recommendations G.711 and G.721. The exact references are given in 5.3.6. Additional application-specific annotation types supported by the ASN.1 type ANY are subject to registration as defined in 6.2.

5.3 Syntax entities of the IIF-DF

In the following, ASN.1 code is indicated by `courier font`. All syntax rules are preceded by a semantics statement. Some rules are succeeded by constraints statements. The rules are ordered in prefix form, with the exception of image attributes, which are described after the non-image data types. The syntax rules, as well as the related semantics and constraints, are divided into the following subclauses:

5.3.1 Entities for the description of the entire IIF-DF

IIF module declaration *IIFDataFormat*
 IIF syntax entity No. 001 *FullDataFormat*
 IIF syntax entity No. 002 *FormatDescriptor*
 IIF syntax entity No. 003 *Version*
 IIF syntax entity No. 004 *Profile*
 IIF syntax entity No. 005 *ContentsHeader*
 IIF syntax entity No. 006 *CharacterString*
 IIF syntax entity No. 007 *SpecialCharacterString*
 IIF syntax entity No. 008 *Contents*
 IIF syntax entity No. 009 *ContentsElement*
 IIF syntax entity No. 010 *ContentsBody*

5.3.2 Entities for the description of images

IIF syntax entity No. 101 *Image*
 IIF syntax entity No. 102 *ImageStructure*
 IIF syntax entity No. 103 *CompoundImageStructure*
 IIF syntax entity No. 104 *CompoundImageArray*
 IIF syntax entity No. 105 *Dimensionality*
 IIF syntax entity No. 106 *DimensionDescription*
 IIF syntax entity No. 107 *Identifier*
 IIF syntax entity No. 108 *Serialization*
 IIF syntax entity No. 109 *DataPlacement*
 IIF syntax entity No. 110 *CompoundImageRecord*
 IIF syntax entity No. 111 *RecordComponent*
 IIF syntax entity No. 112 *CompoundImageList*
 IIF syntax entity No. 113 *CompoundImageSet*
 IIF syntax entity No. 114 *FundamentalImageStructure*
 IIF syntax entity No. 115 *BandRecord*
 IIF syntax entity No. 116 *BandRecordComponent*
 IIF syntax entity No. 117 *MetricArray*
 IIF syntax entity No. 118 *MetricArrayElement*
 IIF syntax entity No. 119 *PixelStructure*
 IIF syntax entity No. 120 *ElementaryPixelStructure*
 IIF syntax entity No. 121 *PixelBandRecord*
 IIF syntax entity No. 122 *PixelBandRecordComponent*

5.3.3 Entities for the description of the representation of pixel values

IIF syntax entity No. 201 *ReferencedUnit*
IIF syntax entity No. 202 *DataUnit*
IIF syntax entity No. 203 *SubdividedDataUnit*
IIF syntax entity No. 204 *SingleDataUnit*
IIF syntax entity No. 205 *BuiltinEncodedDataUnit*
IIF syntax entity No. 206 *BuiltinValue*
IIF syntax entity No. 207 *ComplexValue*
IIF syntax entity No. 208 *ExternallyDefinedDataUnit*
IIF syntax entity No. 209 *CompressedDataUnit*
IIF syntax entity No. 210 *RegisteredDataUnit*
IIF syntax entity No. 211 *ExternalReference*
IIF syntax entity No. 212 *ExternalAddress*

5.3.4 Entities for the description of image-related data

IIF syntax entity No. 301 *ImageRelatedData*
IIF syntax entity No. 302 *Histogram*
IIF syntax entity No. 303 *PartitionClass*
IIF syntax entity No. 304 *LookUpTable*
IIF syntax entity No. 305 *RegionOfInterest*
IIF syntax entity No. 306 *BooleanArray*
IIF syntax entity No. 307 *Ellipse*
IIF syntax entity No. 308 *IntervalND*
IIF syntax entity No. 309 *Interval1D*
IIF syntax entity No. 310 *CoordinateND*
IIF syntax entity No. 311 *SetOfCoordinates*
IIF syntax entity No. 312 *NeighbourhoodArray*
IIF syntax entity No. 313 *IndexND*
IIF syntax entity No. 314 *StaticArray*
IIF syntax entity No. 315 *FeatureList*
IIF syntax entity No. 316 *CoordinateAndFeature*
IIF syntax entity No. 317 *ValueBoundsCollection*
IIF syntax entity No. 318 *TransformationMatrix*
IIF syntax entity No. 319 *PixelRecord*
IIF syntax entity No. 320 *PixelRecordComponent*
IIF syntax entity No. 321 *Tuple*

5.3.5 Entities for the description of image attributes

IIF syntax entity No. 401 *ImageAttribute*
IIF syntax entity No. 402 *MetricDescription*
IIF syntax entity No. 403 *DimensionMapping*
IIF syntax entity No. 404 *MeasurementUnit*
IIF syntax entity No. 405 *DeltaVector*
IIF syntax entity No. 406 *MetricTransformation*
IIF syntax entity No. 407 *Domain*
IIF syntax entity No. 408 *ChannelCharacteristics*

IIF syntax entity No. 409 *CompandorDescription*
 IIF syntax entity No. 410 *ColourRepresentation*
 IIF syntax entity No. 411 *StandardizedSpace*
 IIF syntax entity No. 412 *CIEXYZSpace*
 IIF syntax entity No. 413 *CIEYxySpace*
 IIF syntax entity No. 414 *CIEUVWSpace*
 IIF syntax entity No. 415 *CIEYuvSpace*
 IIF syntax entity No. 416 *CIELabSpace*
 IIF syntax entity No. 417 *CIELuvSpace*
 IIF syntax entity No. 418 *CIEXYZCoordinate*
 IIF syntax entity No. 419 *LinearRGBSpace*
 IIF syntax entity No. 420 *GammaRGBSpace*
 IIF syntax entity No. 421 *YIQColourSpace*
 IIF syntax entity No. 422 *YUVColourSpace*
 IIF syntax entity No. 423 *YCbCrColourSpace*
 IIF syntax entity No. 424 *NonStandardizedSpace*
 IIF syntax entity No. 425 *NonStandardizedRGB*
 IIF syntax entity No. 426 *NonStandardizedIHS*
 IIF syntax entity No. 427 *Primaries*
 IIF syntax entity No. 428 *CIExyCoordinate*
 IIF syntax entity No. 429 *NonStandardizedCMY*
 IIF syntax entity No. 430 *NonStandardizedCMYK*
 IIF syntax entity No. 431 *NonStandardizedNBand*
 IIF syntax entity No. 432 *ColourBand*
 IIF syntax entity No. 433 *TestColour*
 IIF syntax entity No. 434 *PIKSControl*

5.3.6 Entities for the description of image annotations

IIF syntax entity No. 501 *ImageAnnotation*
 IIF syntax entity No. 502 *Location*

5.3.7 Entities for the description of basic data objects

IIF syntax entity No. 601 *BasicDataObject*
 IIF syntax entity No. 602 *BasicDataType*
 IIF syntax entity No. 603 *CompoundDataType*
 IIF syntax entity No. 604 *BasicArray*
 IIF syntax entity No. 605 *BasicRecord*
 IIF syntax entity No. 606 *BasicRecordComponent*
 IIF syntax entity No. 607 *BasicList*
 IIF syntax entity No. 608 *BasicSet*
 IIF syntax entity No. 609 *ElementaryDataType*

5.3.1 Entities for the description of the entire IIF-DF

IIF module declaration

IIFDataFormat

Semantics

IIFDataFormat is the name of the IIF-DF module. Besides the full syntax (given by the *FullDataFormat* entity), the module also exports the *Image*, *ImageRelatedData*, *ImageAttribute*, *ImageAnnotation*, and *BasicDataObject* entities. This provides other ASN.1-notated applications with direct access to these sub-objects.

Two entities are imported from other modules: the *PixelFieldEncoding* entity is imported from ISO/IEC 12089 for the efficient representation of pixel fields and the *DOR* entity is imported from ISO/IEC 10031 for the representation of *Distinguished Object References* (DOR). For a description of the DOR, refer to 5.3.3

In order to obtain the full syntax for the module specification, the term <<declarations>> needs to be replaced with the syntax portions of all subsequent syntax entities within 5.3.

The *FullDataFormat* entity is described here in 5.3.1. Its subentities refer to all other subclauses of 5.3. The *Image* entity is described in 5.3.2. Its subentities refer to 5.3.3 (for the placement of pixel data) and 5.3.4, 5.3.5, and 5.3.6 (for image-related data, attributes and annotations that are directly attached to image objects). The *ReferencedUnit*, *ImageRelatedData*, *ImageAttribute*, *ImageAnnotation* and *BasicDataObject* entities are described in 5.3.3, 5.3.4, 5.3.5, 5.3.6, and 5.3.7, respectively.

Syntax

```
IIFDataFormat {iso standard 12087 iif (3) df (0)} DEFINITIONS ::=
BEGIN

EXPORTS

    FullDataFormat,                -- No. 001
    Image,                        -- No. 101
    ReferencedUnit,               -- No. 201
    ImageRelatedData,             -- No. 301
    ImageAttribute,               -- No. 401
    ImageAnnotation,              -- No. 501
    BasicDataObject;              -- No. 601

IMPORTS

    PixelFieldEncoding FROM
    IIFEncoding {iso standard 12089 iif-encoding (1)}
    DOR FROM
    DOR-definitions {joint-iso-ccitt 10031 reference-definition (0)};

<<declarations>>

END
```

IIF syntax entity No. 001*FullDataFormat*

and

IIF syntax entity No. 002*FormatDescriptor***Semantics**

The *FullDataFormat* entity is the root of the entire IIF-DF syntax. It contains the *format-descriptor*, *contents-header*, and *contents* components.

The *FormatDescriptor* entity contains some meta-information: a *self-identification* component that carries the official identifier of the IPI-IIF syntax, the *profile* component which allows for setting a conformance profile and the *version* component that allows for a textual description of the syntax version.

All image data, image-related data, image annotations, and image attributes are contained in the *contents* component. The *ContentsHeader* entity provides some basic text fields to verbally describe the contents.

Syntax

```
FullDataFormat ::= [APPLICATION 0] IMPLICIT SEQUENCE
{
  format-descriptor    [0] IMPLICIT FormatDescriptor,      -- No. 002
  contents-header      [1] IMPLICIT ContentsHeader OPTIONAL, -- No. 005
  contents             [2] IMPLICIT Contents                -- No. 008
}
```

```
FormatDescriptor ::= SEQUENCE
{
  self-identification  [0] IMPLICIT OBJECT IDENTIFIER DEFAULT
                        { iso standard 12087 iif (3) df (0) },
                                                                -- No. 706
  version              [1] IMPLICIT Version OPTIONAL,        -- No. 003
  profile              [2] IMPLICIT Profile OPTIONAL          -- No. 004
}
```

Constraints

If the *Profile* entity is missing, a full syntax without any further constraint set is assumed.

IIF syntax entity No. 003***Version*****Semantics**

The *Version* entity stands for a textual description of the syntax version to which the given data stream conforms. The name of the syntax version is given by the *standard* component and the publication date of the syntax (either the publication date of ISO/IEC 12087-3 or the publication date of a revision of ISO/IEC 12087-3) is given by the *publication-date* component.

NOTE - The specification method for the *publication-date* component conforms to the method that is used within the ISO/IEC 8613 (ODA/ODIF).

Syntax

```
Version ::= SEQUENCE
{
  standard          [0] IA5String DEFAULT "ISO/IEC 12087-3",    -- No. 722
  publication-date   [1] IA5String                               -- No. 722
}
```

Constraints

None.

IIF syntax entity No. 004***Profile*****Semantics**

The *Profile* entity stands for the description of profiles, specifying that the IIF-DF is restricted to a certain subset. One of the following predefined conformance profiles may be chosen: *full profile*, *full PIKS profile*, and *foundation profile*.

For a definition of these profiles refer to clause 6. Additional application profiles are subject to registration as defined in 6.2.

Syntax

Profile ::= IA5String

-- No. 702

Constraints

For the *Profile* entity, only the values "*full profile*", "*full PIKS profile*", and "*foundation profile*" and values which have been internationally registered are permitted. Refer to 6.2.

IIF syntax entity No. 005***ContentsHeader*****Semantics**

The *ContentsHeader* entity provides some common information about the contents of the IIF data stream. No further semantics are defined for the convention of the *title*, *owner*, *date-and-time*, and *message* components. Also, no semantics are defined for the *application-data* component which can be represented using any ASN.1 type.

NOTE - Information fields, such as "processing platform" or "acquisition process" are regarded as too application-specific to be included in this header as a separate entity. This kind of information can either be put into the message field as readable text or handled as an image annotation or image attribute (using either one of the built-in fields or a freeform field).

EXAMPLE - The *application-data* component (as well as any other component that is typed with the ASN.1 type ANY) could be structured by an application in the following way:

```
ApplicationData ::= SET OF
{
  SEQUENCE
  {
    tag      [0] IMPLICIT INTEGER,
    value    [1] IMPLICIT OCTET STRING
  }
}
```

Syntax

```
ContentsHeader ::= SEQUENCE
{
  title           [0] CharacterString OPTIONAL,      -- No. 006
  owner           [1] CharacterString OPTIONAL,      -- No. 006
  date-and-time   [2] GeneralizedTime OPTIONAL,     -- No. 724
  message         [3] CharacterString OPTIONAL,      -- No. 006
  application-data [4] ANY OPTIONAL
}
```

Constraints

None.

IIF syntax entity No. 006*CharacterString*

and

IIF syntax entity No. 007*SpecialCharacterString***Semantics**

The *CharacterString* entity is mainly meant for the description of a text readable by a person. It provides two alternatives. The *standard-characters* component is used for the representation of ASCII characters according to the International Alphabet No. 5. The *special-characters* component is used for all other character sets.

The *SpecialCharacterString* entity supports all different kinds of national and international character sets. The *character-set-escape* component is used to indicate the character set in use. This is done via the internationally standardized escape sequences according to ISO 2022. The *characters* component contains the character string.

Syntax

CharacterString ::= [APPLICATION 1] CHOICE

```
{
  standard-characters    [0] IMPLICIT IA5String,           -- No. 722
  special-characters     [1] IMPLICIT SpecialCharacterString -- No. 007
}
```

SpecialCharacterString ::= SEQUENCE

```
{
  character-set-escape  [0] IMPLICIT OCTET STRING,         -- No. 704
  characters             [1] IMPLICIT OCTET STRING          -- No. 704
}
```

Constraints

None.

IIF syntax entity No. 008*Contents*

and

IIF syntax entity No. 009*ContentsElement***Semantics**

The *Contents* entity consists of a sequence of *ContentsElement* entities.

The *ContentsElement* entity provides a sequence of *prolog*, *body*, and *epilog* components. The optional *prolog* and *epilog* components do not carry any semantics in the context of ISO/IEC 12087. The only reason for their presence in this syntax entity in combination with the recursive definition of the *ContentsBody* entity is to prepare the introduction of a segmentation concept, associated with type definitions and segment-specific attributes which may be given in a separate Amendment to ISO/IEC 12087.

Syntax

Contents ::= SEQUENCE OF *ContentsElement*

ContentsElement ::= SEQUENCE

```
{
  prolog    [0] ANY OPTIONAL,
  body      [1] IMPLICIT SEQUENCE OF ContentsBody,
  epilog    [2] ANY OPTIONAL
}
```

-- No. 010

Constraints

The *body* component shall contain only one *ContentsBody* entity.

IIF syntax entity No. 010***ContentsBody*****Semantics**

The *ContentsBody* entity stands for the description of iconic and non-iconic data. The following components may be selected:

- the *image* component contains image structure information and associated pixel data;
- the *image-related data* component contains data that conform to one of the image-related data types as defined in ISO/IEC 12087-1.
- the *image-attributes* component contains data that conform to one of the attribute types as defined in ISO/IEC 12087-1.
- the *image-annotations* component contains data that conform to one of the attribute types as defined in ISO/IEC 12087-1.
- the *basic-data-component* contains data that are structured according to a basic data type as defined in ISO/IEC 12087-1.

Image annotations and image-related data are provided in two ways: bound to images, using the corresponding subentities within the *Image* entity, or separate from images using the *ContentsBody* entity as stated above. The latter way allows for the exchange of non-iconic parameters from and to the IPI-PIKS without any image data.

Syntax

```
ContentsBody ::= CHOICE
{
  image                [0] Image,                -- No. 101
  image-related-data   [1] ImageRelatedData,      -- No. 301
  image-attribute      [2] ImageAttribute,        -- No. 401
  image-annotation     [3] ImageAnnotation,       -- No. 501
  basic-data-object    [4] BasicDataObject        -- No. 601
}
```

Constraints

None.

5.3.2 Entities for the description of images

IIF syntax entity No. 101

Image

Semantics

The *Image* entity consists of the components *image-structure* and *image-data*.

The *image-structure* component stands for the description of the structure of one or more image(s) that is/are contained in the *Image* entity. The structure can conform either to a compound or fundamental image data type. It is expressed as a tree whose root is the *ImageStructure* entity and whose leaves represent elementary data types.

The *image-data* component contains a sequence of data fields to which the image structure sub-components refer. The data fields are called the *ReferencedUnit*.

Syntax

```
Image ::= SEQUENCE
{
  image-structure      [0] IMPLICIT ImageStructure,          -- No. 102
  image-data           [1] IMPLICIT SEQUENCE OF ReferencedUnit -- No. 201
}
```

Constraints

Refer to constraints related to the *ReferencedUnit* entity and the *DataPlacement* entity.

IIF syntax entity No. 102***ImageStructure*****Semantics**

The *ImageStructure* entity stands for the description of IPI-IIF data types as defined in ISO/IEC 12087-1. The image data type can be either a *compound-structure* or a *fundamental-structure*.

NOTES

- 1 The following fields of the "representation attribute", defined in ISO/IEC 12087-1, are represented in the IIF-DF by the *ImageStructure* entity and its subentities: *image size*, *band data type*, *image structure code*.
- 2 This entity and its subentities do not contain pixel values. They are used to specify the structure of image data according to the data types defined in ISO/IEC 12087-1. (The pixel values are contained in the *ReferencedUnit* entities.)

The semantical distinction between compound image structures and fundamental image structures is as follows:

- A compound image structure consists of arrays, records, or lists, where the dimensions of the arrays do not refer to a coordinate space that shall be used for data presentation. The leaves of a compound image structure express the structure of fundamental images.
- A fundamental image structure consists of arrays and records, where the dimensions of all arrays refer to the same coordinate space, and the record components refer to the bands (of a multi-band image). The arrays of the bands may differ in size and element structure but not in dimensionality. The leaves of a fundamental image structure express the type of elementary pixel values.

An arbitrary number of attributes and image-related data may be attached, using the *image-attributes* and *image-related-data* components. Image annotations may be attached using the *image-annotations* component. The *processing-history* component serves as a textual description of the image's processing history.

Syntax

```
ImageStructure ::= SEQUENCE
{
    structure-type CHOICE
    {
        compound-structure      [4] CompoundImageStructure,      -- No. 103
        fundamental-structure  [5] FundamentalImageStructure      -- No. 114
    },
    image-attributes            [0] IMPLICIT SEQUENCE OF
        ImageAttribute OPTIONAL,      -- No. 401
    image-related-data          [1] IMPLICIT SEQUENCE OF
        ImageRelatedData OPTIONAL,    -- No. 301
    image-annotations           [2] IMPLICIT SEQUENCE OF
        ImageAnnotation OPTIONAL,     -- No. 501
    processing-history           [3] IMPLICIT CharacterString OPTIONAL -- No. 006
}
```

Constraints

None.

IIF syntax entity No. 103***CompoundImageStructure*****Semantics**

The *CompoundImageStructure* entity stands for the description of compound image data types, as defined in ISO/IEC 12087-1. It can be an array, a record, a list, or a set.

Syntax

```
CompoundImageStructure ::= CHOICE
{
  compound-image-array      [0] CompoundImageArray,      -- No. 104
  compound-image-record     [1] CompoundImageRecord,      -- No. 110
  compound-image-list       [2] CompoundImageList,        -- No. 112
  compound-image-set        [3] CompoundImageSet          -- No. 113
}
```

Constraints

None.

IIF syntax entity No. 104***CompoundImageArray*****Semantics**

The *CompoundImageArray* entity stands for the description of a compound image data type as defined in ISO/IEC 12087-1. The dimensionality of the array, including dimension identifiers and index ranges for every dimension is given by the *dimensionality* component. The sequential order of the array elements is determined within the *serialization* component. The placing of the data values that are associated with this array description is given within the *data-placement* component. The structure of the array elements is given by the *element-structure* component.

NOTE - The *CompoundImageArray* entity contains the *ImageStructure* entity. The *ImageStructure* entity, in turn, contains the *CompoundImageArray*, *CompoundImageRecord*, *CompoundImageList*, and *CompoundImageSet* entities. With this syntactical, recursive construction, image data types of arbitrary complexity may be constructed.

Syntax

```
CompoundImageArray ::= SEQUENCE
{
    dimensionality          [0] IMPLICIT Dimensionality,      -- No. 105
    serialization           [1] IMPLICIT Serialization,        -- No. 108
    data-placement          [2] IMPLICIT DataPlacement,        -- No. 109
    element-structure       [3] IMPLICIT ImageStructure        -- No. 102
}
```

Constraints

None.

III syntax entity No. 105***Dimensionality*****Semantics**

The *Dimensionality* entity contains the array dimensionality given by the *number-of-dimensions* component. For every dimension, a description, given by the *dimension-descriptions* component, is required.

Syntax

```
Dimensionality ::= SEQUENCE
{
    number-of-dimensions      [0] IMPLICIT INTEGER (1..MAX),      -- No. 702
    dimension-descriptions    [1] IMPLICIT SEQUENCE OF
                                DimensionDescription -- No. 106
}
```

Constraints

The number of *DimensionDescription* entities contained in the sequence shall equal the value of the *number-of-dimensions* component.

IIF syntax entity No. 106***DimensionDescription*****and****IIF syntax entity No. 107*****Identifier*****Semantics**

The *DimensionDescription* entity contains a description of one of the array's dimensions. It consists of the *dimension-identifier*, *lower-boundary*, and *upper-boundary* components. The latter two components give the range of the dimension in terms of a lower and an upper boundary. The *upper-boundary* component is optional in order to support an indefinite dimension description. The reader is referred to the *Serialization* entity for the constraints that apply for the indefinite form.

EXAMPLE - An indefinite dimension description may be used to create a moving image consisting of a sequence of frames, where the total number of frames is not known at the time the first frame is being created.

The *Identifier* entity allows for the representation of a dimension identifier in terms of the ASN.1 elementary type IA5String. It is used to associate dimension-specific attributes with image dimensions having the same identifier. The following identifiers have predefined semantics as described in ISO/IEC 12087-2:

"x"	horizontal dimension
"y"	vertical dimension
"z"	depths dimension
"t"	temporal dimension
"b"	spectral dimension

NOTE - The *Identifier* entity is also used by other entities, such as *BandIdentifier* and *ComponentIdentifier*.

Syntax

```

DimensionDescription ::= SEQUENCE
{
  dimension-identifier [0] IMPLICIT Identifier,          -- No. 107
  lower-boundary       [1] IMPLICIT INTEGER (0..MAX)
                      DEFAULT 0,                          -- No. 702
  upper-boundary       [2] IMPLICIT INTEGER (0..MAX) OPTIONAL -- No. 702
}

Identifier ::= IA5String                                -- No. 722

```

Constraints

The number of *DimensionDescription* entities contained in the sequence shall equal the value of the *number-of-dimensions* component in the *Dimension* entity.

The value of the *upper-boundary* component shall not be smaller than the value of the *lower-boundary* component.

See also *Serialization* entity for additional constraints.

IIF syntax entity No. 108***Serialization*****Semantics**

The *Serialization* entity determines the sequential order of array elements. The first descriptor indicates the outer-most loop over array indices. The last descriptor indicates the inner-most loop.

EXAMPLE - Let z be the first descriptor in the IIF data stream with an index range of [1..10], followed by y with an index range of [1..1280] and x with [1..1024]. Then the associated array elements $p(z,y,x)$ have to be interpreted as:

$$p(1,1,1), p(1,1,2), \dots, p(1,1,1024), p(1,2,1), p(1,2,2), \dots, \dots, p(10,1280,1023), p(10,1280,1024)$$
Syntax

Serialization ::= SEQUENCE OF Identifier

-- No. 107

Constraints

The number of *Identifier* entities contained in the sequence shall equal the dimensionality of the array to which the serialization entity is attached. The identifier values shall be identical to those declared within the *dimensionality* component that is attached to the array mentioned above.

Only the first *Identifier* entity may have the value of a *dimension-identifier* component which is declared as being infinite within the *DimensionDescription* entity.

IIF syntax entity No. 109

DataPlacement

Semantics

The *DataPlacement* entity determines whether pixel values that belong to a certain image structure description are stored all together within one *ReferencedUnit* entity, or whether they are split up into multiple *ReferencedUnit* entities.

The structure that describes an image may be regarded as a tree whose root is the *ImageStructure* entity. All other entities that further specify the substructure (*CompoundImageArray*, *CompoundImageRecord*, *CompoundImageList*, *CompoundImageSet*, *MetricArray*, and *BandRecord*) form nodes within this tree. The *ElementaryPixelStructure* entities form the leaves of this tree. At any node within this tree, it may be decided to store all pixel values that belong to this node within one *ReferencedUnit* entity.

Three alternatives are provided for the pixel values that belong to the "node" to which this *DataPlacement* entity is attached. The alternatives are associated with the following integer values:

- distributed* (1): The pixel values that belong to this node are split up into multiple *ReferencedUnit* entities, each of which is associated with one of the substructures of this node.
- combined* (2): The pixel values that belong to this node are stored all together within one *ReferencedUnit* entity, associated with this level of the image structure tree.
- included* (3): The pixel values that belong to this node are stored, together with pixel values belonging to other nodes on the image structure tree, as part of a *ReferencedUnit* associated with a higher level node.

EXAMPLE - Given an image structure that consists of a tiled image, expressed as a two-dimensional *MetricArray* of two-dimensional *MetricArrays* of integer pixels.

Then, a "1" on the level of the upper array and a "2" on the level of the lower array indicate that there exist *ReferencedUnit* entities for every tile, each of which contains all pixel values that belong to this tile.

On the other hand, a "2" on the upper level array and (necessarily) a "3" on the lower level array indicate that all pixel values of the whole image are stored within a single *ReferencedUnit* entity. In this case, the *ReferencedUnit* entity consists of a data stream which first contains all pixel values of the first tile, followed by the pixel values of the second tile, etc. There are no delimiters in the data stream that indicate the tile boundaries. Since the upper array is two-dimensional, a tile represents an object in the two-dimensional space. Thus, a serialization needs to be given in order to unambiguously determine, which tile is stored at what position in the data sequence. This is given by the *serialization* component of the upper array description.

NOTE - Depending on the values for the *DataPlacement* entity, references need to be included in the *ReferencedUnit* entities to indicate which *ReferencedUnit* belongs to which node in the image structure tree. For the description of the *ReferencedUnit* entity, refer to 5.3.3.

Syntax

```
DataPlacement ::= INTEGER -- No. 702
                {
                distributed(1),
                combined(2),
                included(3)
                }
```

Constraints

For the *DataPlacement* entity, only values in the range of (1..3) and values which have been internationally registered are permitted. Refer to 6.2.

On every path from the root of the image structure tree to a leaf there shall be one and only one node that is marked with a "2." All nodes located above shall be marked with "1" and all nodes located below shall be marked with "3."

IIF syntax entity No. 110***CompoundImageRecord*****and****IIF syntax entity No. 111*****RecordComponent*****Semantics**

The *CompoundImageRecord* entity stands for the description of a compound data type, as defined in ISO/IEC 12087-1. The number of record components is given by the *number-of-components* component. Then the placement of the data that are associated with the record is given by the *data-placement* component. The structure of each record component is given by the *record-components* component.

The *RecordComponent* entity consists of an identifier, called the *component-identifier*, and the structure, called the *component-structure*.

NOTE - The *CompoundImageRecord* entity contains the *ImageStructure* entity. The *ImageStructure* entity, in turn, contains the *CompoundImageArray*, *CompoundImageRecord*, *CompoundImageList*, and *CompoundImageSet* entities. With this syntactical recursive construction, image data structures of arbitrary complexity may be constructed.

Syntax

CompoundImageRecord ::= SEQUENCE

```
{
  number-of-components  [0] IMPLICIT INTEGER (1..MAX),      -- No. 702
  data-placement        [1] IMPLICIT DataPlacement,        -- No. 109
  record-components     [2] IMPLICIT SEQUENCE OF
                        RecordComponent                      -- No. 111
}
```

RecordComponent ::= SEQUENCE

```
{
  component-identifier  [0] IMPLICIT Identifier,            -- No. 107
  component-structure   [1] IMPLICIT ImageStructure         -- No. 102
}
```

Constraints

The number of *RecordComponent* entities contained in the following sequence shall equal the value of the *number-of-components* entity.

The character strings of the *component-identifier* entities shall be different from each other in order to allow for unambiguous referral to record components.

The character strings of the *component-identifier* entities shall not contain the characters "/" and "." because these characters have a special meaning for the *reference-label* components within the *ReferencedUnit* entities.

IIF syntax entity No. 112***CompoundImageList*****Semantics**

The *CompoundImageList* entity stands for the description of a compound data type, as defined in ISO/IEC 12087-1. The number of list elements is given by the *number-of-elements* component. This component is optional in order to support lists whose length is indefinite at the beginning of the encoding process.

The placement of the data that are associated with the list is given by the *data-placement* component. The structure of the list elements is given by the *element-structure* entity.

NOTES

1 The *element-structure* component only contains a description of the data structure of the list elements but not the elements themselves. Since lists are homogeneous, one *ImageStructure* entity is sufficient to describe the entire list. The list elements are stored in *ReferencedUnit* entities according to the data placement described by the *data-placement* component.

2 The *CompoundImageList* entity contains the *ImageStructure*. The *ImageStructure* entity, in turn, contains the *CompoundImageArray*, *CompoundImageRecord*, *CompoundImageList*, and *CompoundImageSet* entities. With this syntactical, recursive construction, image structures of arbitrary complexity may be constructed.

Syntax

CompoundImageList ::= SEQUENCE

```
{
  number-of-elements    [0] IMPLICIT INTEGER (1..MAX) OPTIONAL, -- No. 702
  data-placement        [1] IMPLICIT DataPlacement,             -- No. 109
  element-structure     [2] IMPLICIT ImageStructure              -- No. 102
}
```

Constraints

None.

IIF syntax entity No. 113***CompoundImageSet*****Semantics**

The *CompoundImageSet* entity stands for the description of a compound data type, as defined in ISO/IEC 12087-1. The number of set elements is given by the *number-of-members* component. This component is optional in order to support sets whose cardinality is indefinite at the beginning of the encoding process.

The placement of the data that are associated with the set is given by the *data-placement* component. The structure of the set members is given by the *member-structure* entity.

NOTES

1 The *member-structure* component only contains a description of the data structure of the set members but not the members themselves. Since sets are homogeneous, one *ImageStructure* entity is sufficient to describe the entire set. The set members are stored in *ReferencedUnit* entities according to the data placement, described by the *data-placement* component.

2 The *CompoundImageSet* entity contains the *ImageStructure*. The *ImageStructure* entity, in turn, contains the *CompoundImageArray*, *CompoundImageRecord*, *CompoundImageList*, and *CompoundImageSet* entities. With this syntactical, recursive construction, image structures of arbitrary complexity may be constructed.

Syntax

CompoundImageSet ::= SEQUENCE

```
{
  number-of-members    [0] IMPLICIT INTEGER (1..MAX) OPTIONAL, -- No. 702
  data-placement       [1] IMPLICIT DataPlacement,             -- No. 109
  member-structure     [2] IMPLICIT ImageStructure              -- No. 102
}
```

Constraints

None.

IIF syntax entity No. 114***FundamentalImageStructure*****Semantics**

The *FundamentalImageStructure* entity stands for the structural description of fundamental image data type, as defined in ISO/IEC 12087-1. It can be an n-dimensional array of pixel values (given by the *metric-array* component) or a record of bands (given by the *band-record* component) each of which is defined as an n-dimensional array of pixel values.

All bands of a *FundamentalImageStructure* entity are assumed to refer to the same coordinate space. Hence, the *FundamentalImageStructure* entity may comprise a collection of pixel arrays that represent slices of a multi-spectral image or a simple colour image but it may not comprise a collection of pixel arrays that have been acquired at different locations or show different objects. The latter kind of compound image structure needs to be modelled with the *CompoundImageStructure* entity.

NOTES

- 1 Besides other image types, the *FundamentalImageStructure* entity covers elementary image data types, as defined in ISO/IEC 12087-1. For the definition of the relation between compound, fundamental, and elementary image types, refer to ISO/IEC 12087-1.
- 2 Because the *FundamentalImageStructure* entity allows multi-band images to be described, these images need not be modelled as compound images.

Syntax

```
FundamentalImageStructure ::= CHOICE
{
  band-record      [0] BandRecord,           -- NO. 115
  metric-array     [1] MetricArray           -- NO. 117
}
```

Constraints

None.

IIF syntax entity No. 115***BandRecord*****Semantics**

The *BandRecord* entity stands for the description of a record of image bands, each of which is identified by its component identifier. The number of bands is given by the *number-of-bands* component. The placement of the pixel values that belong to this structural description is determined by the *data-placement* component. The structure of each band is given by the *record-components* component.

Because the pixel values belonging to an image structure definition are ordered according to the sequence in which the structural definition of the image is given, the level at which an image is split into bands has an influence on the sequential order of pixel values.

EXAMPLE - An image that is described as a *metric-array* of *band-records* of three pixels forms a pixel-interleaved image whereas an image that is described as a *band-record* of three *metric-arrays* of pixels forms a band-interleaved image.

NOTE - The *BandRecord* entity is organized in conformance with the *CompoundImageRecord* entity. The term "band" as part of the name of the *BandRecord* entity reinforces the fact that the record components always have the meaning of bands of a multi-band image.

Syntax

```
BandRecord ::= SEQUENCE
{
  number-of-bands      [0] IMPLICIT INTEGER (1..MAX),      -- No. 702
  data-placement       [1] IMPLICIT DataPlacement,         -- No. 109
  record-components    [2] IMPLICIT SEQUENCE OF
                        BandRecordComponent                 -- No. 116
}
```

Constraints

The number of *BandRecordComponent* entities contained in the *record-components* component shall equal the value of the *number-of-bands* entity.

IIF syntax entity No. 116***BandRecordComponent*****Semantics**

The *BandRecordComponent* entity stands for the description of an image band which is part of a multi-band image, described by the *BandRecord* entity. The *BandRecordComponent* consists of the *band-identifier* component which is used to associate attributes describing the nature of the band with this band. Furthermore, it consists of the *component-structure* component which gives the structure of the band.

Syntax

```
BandRecordComponent ::= SEQUENCE
{
  band-identifier          [0] IMPLICIT Identifier,          -- No. 107
  component-structure      [1] IMPLICIT MetricArray          -- No. 117
}
```

Constraints

The character strings of the *band-identifier* entities shall be unique in order to allow unambiguous referral to record components.

The character strings of the *band-identifier* entities shall not contain the characters "/" and "." because these characters have a special meaning for the *reference-label* components within the *ReferencedUnit* entities.

IIF syntax entity No. 117***MetricArray*****Semantics**

The *MetricArray* entity stands for the description of an n-dimensional array. The dimensionality, (including the dimension identifiers and index ranges for every dimension) is given by the *dimensionality* component. The sequential order of the array elements is determined within the *serialization* component. The placing of the data values that are associated with this array description is given within the *data-placement* component. The structure of the array elements is given by the *element-structure* component.

NOTES

1 The structure of the *MetricArray* entity is similar to the structure of the *CompoundImageArray*. The term "metric" as part of the name of the *MetricArray* entity reinforces the fact that the dimensions of this array are assumed to refer to a coordinate space.

2 The term "dimensionality of the *MetricArray* entity" is used in subsequent entities. This always refers to the value of the *number-of-dimensions* component within the *Dimensionality* entity.

Syntax

```

MetricArray ::= SEQUENCE
{
    dimensionality          [0] IMPLICIT Dimensionality,      -- No. 105
    serialization           [1] IMPLICIT Serialization,        -- No. 108
    data-placement          [2] IMPLICIT DataPlacement,        -- No. 109
    element-structure       [3] MetricArrayElement             -- No. 118
}

```

Constraints

None.