INTERNATIONAL STANDARD

ISO/IEC 12087-3

First edition
1995-02-15

AMENDMENT 1
1996-12-15

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

Part 3:
Image Interchange Facility (IIF)

AMENDMENT 1: Type definition, scoping, and logical views for image interchange facility

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

Partie 3: Accessoires pour l'échange d'images (IIF)

AMENDEMENT 1: Définition de type, domaine d'application et vues logiques pour les accessoires pour l'échange d'images (IIF)

Reference number
ISO/IEC 12087-3:1995/Amd.1:1996(E)

AVA Documents
Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

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.

Amendment 1 to International Standard ISO/IEC 12087-3:1995 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee 24, *Computer graphics and image processing*.

© ISO/IEC 1996

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

Documents
Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

Type definition, scoping, and logical views for image interchange facilty

5 The IIF data format (IIF-DF)

.....

5.1 Basic features of the IIF-DF

.....

Add the following new subclause:

5.1.5 Segment Structure of IIF Data Stream

The content of an IIF data stream consists of zero or more segments, hierarchically structuring the data in a tree like manner. Considering ASN.1 constructs as an alphabet, IPI Part3 (IIF) can be seen as a grammar which combines elements of this alphabet to build entities carrying image processing specific semantics as defined in IPI Part 1 and Part 2. In addition to the straightforward usage of these entities, an application may select one, two or more of them, group them together, and associate some additional or new semantics with such a set. Segments provide the mechanisms to make that grouping persistent. Each IIF segment has three parts.

The first part, called *prolog* (entity number 901), serves as the definition space for attributes that apply to the second part, called *body* (entity number 010). The *prolog* provides also facilities to associate a segment with a unique name and user defined label. These can be used as handles while processing the IIF data stream. The *prolog* may also contain a reference to a segment type definition in order to constrain the structure of segment to conform that definition.

The *body* of a segment contains all IPI-CAI data types required to interchange image and image related data as well as types necessary for the application specific structuring of these data.

The third part of a segment, called *epilog* (entity 902), is provided for syntactical and processing reasons. It specifies a mark-up denoting the boundary of a segment, and it may contain useful IIF profile dependent or application specific information to facilitate random access to an IIF data stream residing in the memory buffer or on a file.

The main objectives which are addressed by the introduction of segmentation into IIF can be summarised as follows below.

5.1.5.1 Attribute Inheritance and Management

Each segment may contain a collection of image related data (entity number 301), image attributes (entity number 401), image annotations (entity number 501) and basic data types (entity number 602) referred to as "segment attributes" (entity number 903). These attributes are inherited by the child segments i.e. by segments nested in the given segment. A child segment may in turn specify an attribute which has a higher precedence than the inherited one and so modify it. In this sense, every segment carries a set of attributes, which are either specified in that segment or inherited from the parent segment. These attributes are considered as **default** attributes of the given segment and apply to the content specified in the body of a segment, unless they are overwritten by newly specifying them **immediate** in the body of a segment (refer to syntax entity No. 008, *ContentsBody*).

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

The definition of an attribute allows to specify the type of an attribute and the value of an attribute. The type of an attribute is specified as a hierarchy of context sensitive ASN.1 tags referring to the grammar of IIF data stream. The value of an attribute is specified according to definitions provided in IPI-CAI. Explicit management scheme for instantiation of attributes is outlined by the segment type definition facility (see clause below on constraints on topology and attributes of a segment). This scheme defines the rules of presence and propagation for each attribute specified in the definition of a segment type (via construct *SegmentStructure* of *SegmentTypeDefn* entity number 910). The presence and propagation rule for an attribute is a combination of predicates which shall be applied to the concerned attribute by an application processing the content of a segment (*AttributeOccurrence* entity number 912).

5.1.5.2 Constraints on Topology and Attributes of a Segment

Any segment may be constrained to have a specific topology and a prescribed set of segment attributes associated with this topology. This is achieved by declaring the segment to be of a given "segment type".

The constraints on topology of a segment structure are defined in terms of a nested combination of orderings (sequence, set, choice) and occurrences (one or more required or optional items). A set of attributes can be associated, in the way provided in this specification, with every item of the segments' structure. Thus, the segment type is defined by the specification of constrains on its topology and by the specification of its attributes.

A segment which is constrained to be of a given type must possess the topology and attributes as prescribed in the definition of that type. Note however, that for the attributes specified in a definition of segment type, the value of an attribute, and any part of the definition of an attribute type can remain undefined. See in that context data-required construct in several entities, e.g. IndexND (entity number 308), CompoundDataType (entity number 603) etc. An undefined type and value specification can be completed, without the violation of a segment type, by an application using the segment of a such type. Therefore, in an IIF data stream the segments adhering to the same type can have not only different content, different values of attributes, but also their attributes can differ in some parts of its type definition. Such segments may constitute thereby hierarchy of classes of segment types. Note also that an application can associate with given segment type, or with given class of segment types, specific methods how to process the content of such segments (see the construct user-label in SegmentProlog entity number 901, and similar construct in SegmentTypeDefn entity number 910).

5.1.5.3 Symbolic References

Each segment may contain a collection of definitions, each of which is associated with an identifier. The constructs which can be defined in such a way (with help of the *NamedItems* entity, entity number 904) are image related data, image attributes, image annotations, basic data types, image structures and segment types. In contrast to segment attributes, these constructs do not apply to the content specified in the body of a segment, and are not directly inherited by the child segments. Instead, they are used as targets for references in other segments which may need the same constructs: such segments will rather make a symbolic reference to an appropriate definition than duplicate the same definitions in their headers or bodies.

Reference mechanism is implemented within the name and address space associated with the segments. This space consists of segment identifiers unique in given context. By definition, such a context can be for example a specific IIF data stream, or a referenced external data repository. Note that by merging together IIF data streams, or different external data repositories, the uniqueness of identifiers must be preserved. The technique recommended to achieve this goal is to use the addressing scheme as specified in ISO/IEC 10031, Distributed Office Application Model (DOAM), Part 2: Distinguished Object Reference (DOR) to generate segment identifiers.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

5.1.5.4 Logical Views of Image Data

Instead of physically supplying the image data in its content, a segment may use there symbolic references into the image structure describing some physical data set. Since the image structure fully corresponds to the image data, such references into image structure are equivalent to (i.e. can be resolved to result in) references into image data. A **logical view** of a remote data set is a segment which has symbolic references into parts of image data supplied elsewhere. The referencing mechanism is implemented through the naming of image structures within the name and address space as described above in clause on symbolic references. See also in this context the definition of the IIF syntax entity, *ReferenceUnit* (entity number 201).

As long as the reference path is a-cyclic, the targets of references may themselves be symbolic references, resulting in a mechanism for **logical** reordering of **physical** image data. It is obvious that the logical views required by the application can not go beyond the granularity implied by the physical data set, i.e., the atomic elements a logical view consists of can not be smaller than referable elements of the referenced structure. This implies, that some application may need to restructure the physical data set collected by another application in order to offer a more detailed granularity required by its own semantics.

5.1.5.5 Information Integration Support

An IIF data stream may need to integrate other data. These could be modelled as another IIF data stream, as a flat or structured stream of ASN.1 tokens following rules of a grammar other then IIF, as an arbitrary octet string, or as any other stream of bits. Structuring facilities and mechanisms associated with these facilities allow to differentiate precisely between all three cases providing well defined rules how to access the data in the best way.

External reference mechanism allows that a repository of information can reside outside the IIF data stream. The ASN.1 object identification scheme is used to provide necessary information about the syntax of referenced data. The so called *EntityHandle* (entity number 917) offers a flexible mechanism to choose between different kind of pointers to the data structured according to IIF grammar and to the data encoded according to any ASN.1 or even non-ASN.1 grammar. The syntax of this pointer has well defined semantics within IIF grammar but it is also flexible enough to point into other repositories (e.g. Common Object Request Broker specified by X/Open and Object Management Group). The possibility to type segments introduced by *SegmentTypeDefn* entity provides a facility to bind given application specific processing methods to required parts of the IIF data stream.

5.1.5.6 Access Support

While stored in a file or buffered in the memory of a computer, an IIF data stream usually represents large amount of sequential organised data. Therefore random access to an arbitrary chosen part of these data is, somehow, not trivial problem in terms of time and consumed resources. It can be, however, significantly facilitated by the "a priori" knowledge of generic logical structure for given IIF data stream. Such a generic structure will consist of a hierarchy of segment types definitions, and it can be provided as a type guide in *ContentsHeader* entity, or as an explicit profile definition in *Profile* entity.

Based on possible unique application specific semantics which can be associated with the elements of logical structure, the logical structure will help application to navigate in an IIF data stream. Otherwise, while mapped to a file or to a buffer, the logical structure can directly enable paging mechanism of specific implementation platform as the random access tool for an IIF data stream. The definition of such mapping, however, is outside the scope of this IPI-IIF. Considered to be application dependent, it can be implemented through further specification of *access-information* component of *ContentsHeader* entity and *access-optimizer* component of *SegmentEpilog* entity.

AVA Documents Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

*Replace 5.3 by the following. The syntax entities marked with *) are extended in an upward compatible way. The syntax entities marked with **) are new.*

5.3 Syntax entities of the IIF-DF

In the following, ASN.1 code is indicated by `courier font`. All syntax rules are preceeded by a semantics statement. Some rules are succeeded by constraints statements. The rules are ordered in prefix form, with the exceptions of
1) attributes are described after the non-image data types,
2) segment-related entities (5.3.8 and 5.3.9),
3) reference mechanism entities (5.3.10).

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*

AVA Documents Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

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*)*

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

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*)*

5.3.8 Entities for the description of segmentation facility **)

IIF syntax entity No. 901 *SegmentProlog*
IIF syntax entity No. 902 *SegmentEpilog*
IIF syntax entity No. 903 *SegmentAttributes*

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

5.3.9 Entities for the description of segment types **)

IIF syntax entity No. 904	*NamedItems*
IIF syntax entity No. 905	*ImageStructureDefn*
IIF syntax entity No. 906	*ImageRelatedDataDefn*
IIF syntax entity No. 907	*ImageAttributeDefn*
IIF syntax entity No. 908	*ImageAnnotationDefn*
IIF syntax entity No. 909	*BasicDataObjectDefn*
IIF syntax entity No. 910	*SegmentTypesDefn*
IIF syntax entity No. 911	*ExpressionElement*
IIF syntax entity No. 912	*AttributeOccurence*
IIF syntax entity No. 913	*StructureDefn*
IIF syntax entity No. 914	*OccurenceDefn*
IIF syntax entity No. 915	*RepeatElement*
IIF syntax entity No. 916	*StructureElement*

5.3.10 Entities for the description of reference mechanism **)

IIF syntax entity No. 917	*EntityHandle*
IIF syntax entity No. 918	*ImageStructureLabel*
IIF syntax entity No. 919	*ImageRelatedDataLabel*
IIF syntax entity No. 920	*ImageAttributeLabel*
IIF syntax entity No. 921	*ImageAnnotationLabel*
IIF syntax entity No. 922	*BasicDataTypeLabel*
IIF syntax entity No. 923	*SegmentLabel*
IIF syntax entity No. 924	*SegmentTypeLabel*
IIF syntax entity No. 925	*Label*
IIF syntax entity No. 926	*ImageStructureRef*
IIF syntax entity No. 927	*ImageRelatedDataRef*
IIF syntax entity No. 928	*ImageAttributeRef*
IIF syntax entity No. 929	*ImageAnnotationRef*
IIF syntax entity No. 930	*BasicDataTypeRef*
IIF syntax entity No. 931	*SegmentTypeRef*
IIF syntax entity No. 932	*SegmentRef*
IIF syntax entity No. 933	*ExternalRefIndex*
IIF syntax entity No. 934	*ImageDataRef*

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

Replace the syntax entity 004 by the following:

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. While application-profile constraints the conformance-profile by "linguistic" means, assuming that specific name is known to the IIF Gateway, or to the application, the same is achieved by profile-definition, pointing to the formal definition how data types shall be used in the current IIF data stream, or simply, giving a "generic example" of what is allowed inside the conformance-profile. The profile-definition can be therefor considered as generic logical structure of a given IIF data stream. Both application-profile and profile-definition may constrain the range implied by conformance-profile, but never extend it.

Syntax

```
Profile ::= CHOICE
    {
    application-profile   IA5String,                    -- No. 722
    profile-definition    SegmentTypeRef                -- No. 931
    }
```

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. A generic logical structure can be either defined with help of the *profile-definition* component referencing directly to an external data repository containing definition of segment type that models the structure of an IIF data stream, or this definition shall be taken from *type-guide* component in *ContentsHeader* entity

AVA Documents Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

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.

The *external-references* component describes the sources of data which are outside of the current IIF data stream. The *access-information* component provides additional information which will facilitate random access to the content of the current IIF data stream. It points to the specification of an external file in the list of external references. The *type-guide* component is a pointer to an external data repository from which the default attributes are inherited and from which type definitions can be referenced.

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,
   external-references [5] IMPLICIT SEQUENCE OF
                            ExternalReference          OPTIONAL, -- No. 211
   access-information  [6] IMPLICIT ExternalRefIndex   OPTIONAL, -- No. 927
   type-guide          [7] IMPLICIT ExternalReference  OPTIONAL  -- No. 211
   }
```

Constraints

The *type-guide* component shall be present if *profile-definition* component in *Profile* entity does not specify an external data repository. In specification of *EntityHandle* in the *type-guide* component the *segment-handle* option should be applied.

Documents Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

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. Such a sequence is called a segment.

Image segments group content that is differentiated from the surrounding content by a change in processing characteristics. An application may bind to it the application specific semantics.

The *prolog* component is used to store the attributes that apply to the body of this segment and can be inherited in nested segments. In addition the *prolog* components contains type definitions which can be used in the name space of the current segment or referenced by other segments.

The *body* component contains all IPI-CAI data types required to interchange image and image related data as well as types necessary to assure application specific structuring of these data.

The *epilog* component provides a mark-up of the segment boundary which may contain useful information to facilitate random access to the IIF data stream residing in the memory buffer or on a file.

Syntax

```
Contents ::= SEQUENCE OF ContentsElement

ContentsElement ::= SEQUENCE
    {
    prolog    [0] IMPLICIT SegmentProlog OPTIONAL,              -- No. 901
    body      [1] IMPLICIT SEQUENCE OF ContentsBody,            -- No. 010
    epilog    [2] IMPLICIT SegmentEpilog OPTIONAL               -- No. 902
    }
```

Constraints

None.

Documents
Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

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.
- the *contents-element* provides a recursion because its subentities refer back to the ContentsBody entity. According to the constraint given below, this recursion is prohibited. The only reason for incorporating the *contents-element* component into the syntax is to prepare the introduction of a hierarchical type definition and validity space concept that may be given in a separate Ammendment to this International Standard.

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
    contents-element      [5] ContentsElement           -- No. 009
    }
```

Constraints

None.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

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, elementary image structures and image structure reference 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.
- A referenced image structure is a symbolic reference to an image structure defined in the name space of the current segment or in the name space provided by an external address

Syntax

```
ImageStructure ::= SEQUENCE
   {
   structure-type CHOICE
      {
      compound-structure       [4] CompoundImageStructure,      -- No. 103
      fundamental-structure    [5] FundamentalImageStructure    -- No. 114
      referenced-structure     [6] ImageStructureRef            -- No. 926
      },
   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.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

IIF syntax entity No. 201 *ReferencedUnit*

Semantics

The *ReferencedUnit* entity stands for the description of a unit of pixel values that is marked with a label to indicate to which image structure it belongs. The label is given by the *reference-label* component. The pixel values are contained in the *pixel-values* component which is specified either explicitly (DataUnit) or by a reference to another image (ImageDataRef).

As stated in the description of the *DataPlacement* entity, 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.

For the referral mechanism applied within the *reference-label* components, the following rules are specified:

a) Any node in the tree can be identified by the names of all edges that lead from the root to this node. The names of the edges are determined by the image structure description. They vary depending on the type of the nodes:

   a1) For records (given by the *CompoundImageRecord*, *BandRecord*, and *PixelBandRecord* entities), the names of the edges that lead from the record description to one of its components are defined as the names of the record components, given by the *component-identifier* and *band-identifier* components.

   a2) For arrays (given by the *CompoundImageArray* and *MetricArray* entities), the names of the edges that lead from the array description to one of its elements are defined as the multidimensional indices. The index values for every dimension are ordered according to the *serialization* component that is part of the array description. They are separated with "." characters.

   a3) For lists (given by the *CompoundImageList* entity), the names of the edges that lead from the list description to one of its list elements are defined by the sequential position of set members in the data stream, assuming that the first element in the sequence is designated to be number "1" and the successor of the nth element is assigned to be the $n+1$th element.

   a4) For sets (given by the *CompoundImageSet* entity), the names of the edges that lead from the set description to one of its set members are defined by the sequential position of set members in the data stream, assuming that the first element in the sequence is designated to be number "1" and the successor of the nth element is assigned to be the $n+1$th element.

b) In order to create the value of a *reference-label* component, the names of all edges are concatenated, inserting a "/" character in front of every edge name, beginning with a "/" character for the root.

EXAMPLE - Given an image structure that consists of a two-dimensional array of tiles, each of which consists of a record of 3 bands, called "red," "green," and "blue." Then the green band of the second tile in the first row of tiles is identified with the reference label "/1.2/green". The entire tile is identified with "/1.2" and the whole image is identified with "/".

AVA Documents Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

Syntax

```
ReferencedUnit ::= SEQUENCE
   {
   reference-label  [0] IMPLICIT Identifier,                    -- No. 107
   pixel-values
   CHOICE {
     explicit-values   [1] IMPLICIT  DataUnit,                  -- No. 202
     referenced-values [2] IMPLICIT  ImageDataRef               -- No. 934
     }
   }
}
```

Constraints

For every node in an image structure tree that has a *data-placement* component which is marked with a "2" (see *DataPlacement* entity), there shall exist one and only one *ReferencedUnit* entity that contains all pixel values that belong to this node and that contains a *reference-label* component that describes the path of this node.

The representation of the pixel values and the number of values that are contained in the *DataUnit* entity shall match the image structure description this *ReferencedUnit* refers to.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

IIF syntax entity No. 202 *DataUnit*

Semantics

The *DataUnit* entity can either be a single data field without further subdivision, given by the *single-data-unit* component; a data field that is partitioned into a number of equal-sized portions, given by the *subdivided-data-unit* component; a reference to a *ReferencedUnit* entity which is stored remotely, given by the *external-data-unit* component, or a remote data set, which is specified as *entity-index* component by an index into the list of *external-references* in the header of the current IIF data stream..

For the description of the partitioning concept refer to the *SubdividedDataUnit* entity. For the description of the referral scheme to external data sets refer to the *ExternalReference* entity.

Syntax

```
DataUnit ::= CHOICE
    {
    single-data-unit        [0] SingleDataUnit,                 -- No. 204
    subdivided-data-unit    [1] SubdividedDataUnit,             -- No. 203
    external-data-unit      [2] ExternalReference,              -- No. 211
    external-ref-index      [3] ExternalRefIndex                -- No. 927

    }
```

Constraints

See subentities.

Documents Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

IIF syntax entity No. 211 *ExternalReference*

and

IIF syntax entity No. 212 *ExternalAddress*

Semantics

The *ExternalReference* entity supports a referral mechanism to an externally stored data set within a heterogeneous and distributed computing environment.

NOTE - Within ISO/IEC 12087, this mechanism is exclusively used to refer to a field of pixel values that is not present within the current data stream.

The *object-address* entity is used to describe the address of the external object. Both local environments (e.g., file names in a UNIX file system) and heterogeneous, distributed environments (e.g., addresses of objects that may be obtained via ftp) are supported.

The *object-format* component allows for a description of the format of the externally stored data set. This is done by a unique identifier of the grammar. If this component is absent, the external data portion shall be interpreted as a *ReferencedUnit* entity, according to the IIF-DF syntax.

The *object-internal-id* component is an optional component which may be used to identify a specific object part within the object specified by the *object-address* component (e.g., by giving a byte offset that points from the top of the file to the beginning of the valid data). A concrete interpretation of this component may be specified within an application profile. Application profiles are subject to registration, as defined in 6.2.

The *entity-address* component is an optional component which allows to constrain the refernced data to either a specific segment and its nested segments if the referenced data is an IIF data stream, or to an application specific information quanta if referenced data set is structured otherwise.

The *ExternalAddress* entity provides two alternatives for the representation of an external object address. The *structured-address* component provides a unique location and name of the external object, according to ISO/IEC 10031, Distributed Office Application Model (DOAM), Part 2: Distinguished Object Reference (DOR). The *cleartext-address* component may be used for an informal representation. A concrete interpretation of this component may be specified within an application profile. Application profiles are subject to registration, as defined in 6.2.

Syntax

```
ExternalReference ::= SEQUENCE
  {
  object-address       [0] ExternalAddress,                        -- No. 212
  object-format        [1] IMPLICIT OBJECT IDENTIFIER OPTIONAL,-- No. 706
  object-internal-id   [2] IMPLICIT CharacterString   OPTIONAL,-- No. 006
  entity-address       [3] IMPLICIT EntityHandle      OPTIONAL -- No. 917
  }

ExternalAddress ::= CHOICE
  {
  structured-address    [0] DOR,                                   -- No. 802
  cleartext-address     [1] IMPLICIT CharacterString               -- No. 006
  }
```

Constraints

None.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

IIF syntax entity No. 301 *ImageRelatedData*

Semantics

The *ImageRelatedData* entity stands for the description of image-related data, as defined in ISO/IEC 12087-1. The following types of image-related data are provided: match point, look-up table, histogram, region of interest, neighbourhood, and feature list.

The *identifier* component is used to associate the image-related data with those parts of an image structure which have the same identifier.

The *usage* component allows for a verbal description of the way the image-related data type is used.

The types of image-related data named above are given by the *histogram*, *look-up-table*, *region-of-interest*, *neighbourhood-array*, *static array*, *feature-list*, *value-bounds-collection*, *matrix*, *pixel-record*, *tuple*, and *referenced-data* components. Instead of explicitly specifying one of these types, the *ImageRelatedData* entity may also reference the desired type (via an image-related data reference) from the name space of the current segment or from the name space provided by an external address.

The *data-required* component is usable in the specification of a segment type. If this option is chosen in the specification of a segment type, an application can complete, observing all existing constraints, the type and value any other component of the *ImageRelatedData* entity at the time an IIF data stream is created or processed.

Syntax

```
ImageRelatedData ::= CHOICE {
  data-specified  SEQUENCE
  {
   identifier                  [0] IMPLICIT Identifier,        -- No. 107
   usage                       [1] ANY OPTIONAL,
   data-type CHOICE
    {
    histogram                 [2] Histogram,                  -- No. 302
    look-up-table             [3] LookUpTable,                -- No. 304
    region-of-interest        [4] RegionOfInterest,           -- No. 305
    neighbourhood-array       [5] NeighbourhoodArray,         -- No. 312
    static-array              [6] StaticArray,                -- No. 314
    feature-list              [7] FeatureList,                -- No. 315
    value-bounds-collection   [8] ValueBoundsCollection,      -- No. 317
    matrix                    [9] TransformationMatrix,       -- No. 318
    pixel-record             [10] PixelRecord,                -- No. 319
    tuple                    [11] Tuple                       -- No. 321
    },
    referenced-data          [12] ImageRelatedDataRef,        -- No. 927
   },
   data-required             [13] IMPLICIT NULL               -- No. 705
}
```

Constraints

Choosing the *data-required* component outside the definition of segment type will cancel the specification of *ImageRelatedData* entity and may cause unpredictable results.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

IIF syntax entity No. 302 *Histogram*

and

IIF syntax entity No. 303 *PartitionClass*

Semantics

The *Histogram* entity stands for a histogram, as defined in ISO/IEC 12087-1 and used in ISO/IEC 12087-2.

The number of partition classes is given by the *number-of-classes* component. A description of the partition classes' format is given by the entity *class-description*. In general, this is the data type definition of the pixels to be classified. Then, all partition classes and counts are given by the *classes-and-counts* component.

NOTE - This entity allows one to describe partition classes that are associated with images which have compound pixels (e.g., records of elementary data types).

The *PartitionClass* entity stands for the description of a partition class and the corresponding count.

The partition class is given by two sequences of elementary data values. One is called the *lower-boundary* and the other is called the *upper-boundary*. Both sequences together form an interval in an n-dimensional parameter space given by the pixel data type. All values *v* within the interval

$$lower\text{-}boundary <= v < upper\text{-}boundary$$

belong to the partition class except for the upper-most partition class where the value of the *upper-boundary* is included in the interval.

The *data-required* component is usable in the specification of a segment type. If this option is chosen in the specification of a segment type, an application can complete, observing all existing constraints, the type and value any other component of the *Histogram* entity at the time an IIF data stream is created or processed

Syntax

```
Histogram ::= CHOICE
  {
  data-specified SEQUENCE
    {
    number-of-classes    [0] IMPLICIT INTEGER (1..MAX),        -- No. 702
    class-description    [1] BasicDataType,                    -- No. 602
    classes-and-counts   [2] IMPLICIT SEQUENCE OF
                             PartitionClass,                   -- No. 303
    }
  data-required          [3] IMPLICIT NULL                     -- No. 705
  }

PartitionClass ::= SEQUENCE
  {
  lower-boundary    [0] IMPLICIT SEQUENCE OF BuiltinValue,     -- No. 206
  upper-boundary    [1] IMPLICIT SEQUENCE OF BuiltinValue,     -- No. 206
  count             [2] IMPLICIT INTEGER (0..MAX)              -- No. 702
  }
```

Constraints

The value of the *number-of-classes* component shall equal the number of *PartitionClass* entities within the *classes-and-counts* component.

The *class-description* component shall match the pixel data type defined for the *FundamentalImageStructure* entity, with which the histogram is associated.

AVA Documents Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

The number of *BuiltinValue* entities within the *lower-boundary* sequence shall equal the number of elementary data types of which a compound (pixel) data type (defined by the *class-description* component) is constructed. The same condition holds for the number of *BuiltinValue* entities within the *upper-boundary* component.

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

Choosing the *data-required* component outside the definition of segment type will cancel the specification of *Histogram* entity and may cause unpredictable results.

EXAMPLE - Let an image structure consist of an array of pixels that are records of three integers. Then, a histogram associated with this pixel array (via the image-related data entity) has to have a class description identical to the pixel data type. All partition classes contain three integer values as the lower boundary and three integer values as the upper boundary.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

IIF syntax entity No. 304 *LookUpTable*

Semantics

The *LookUpTable* entity stands for the description of a look-up table, as defined in ISO/IEC 12087-1 and used in ISO/IEC 12087-2.

The *number-of-input-dimensions*, *input-data-types*, *number-of-output-dimensions*, and *output-data-types* components describe the look-up table's format.

The *number-of-input-dimensions* component describes the number of independent elementary input values which are used to index the table. The *input-data-types* component describes the data type of these input values as a whole. The *number-of-output-dimensions* component describes the number of independent elementary output values. The *output-data-types* component describes the data type of these output values as a whole. The *number-of-entries* component gives the total number of look-up table entries. All input and output values are given by a sequence of *BuiltinValue* entities, as defined in 5.3.3.

Both *input-data-types* and *output-data-types* are represented by the *BasicDataType* entity which may be either a compound data type or an elementary data type. Input and output vectors may be specified by records, represented by the *BasicRecord* entity. Thus, every component declaration encompasses a component identifier. These identifiers are used to bound colour space declarations to look-up tables.

The *input-data-types* component may contain real and complex as elementary data types. In these cases, an interpolation method needs to be declared. The *interpolation-method* component may be used for a textual description. If it is absent, a linear interpolation function is assumed for integer and real data types; for complex data types, the interpolation method is application-dependent.

NOTE - Because the input data type may also be a compound data type (e.g., a record), the look-up table provides a general mapping of input vectors to output vectors.

Both *input-values* and *output-values* components form sequences of elementary values. Depending on the input and output dimensionality, a table index and a table entry may consist of multiple elementary values. The *i*th index (represented by one or more elementary values) within the *input-values* component is semantically bound to the *i*th entry (represented by one or more elementary values) within the *output-values* component.

EXAMPLES

1.) Let us assume, the output values of a look-up table are to be interpreted as YIQ colour values. This fact may be expressed by the IIF-DF syntax using the following identifier settings:

ImageStructure:	*band-identifier*s within *BandRecord*:	"table1-a" "table1-b" "table1-c"
LookUpTable:	*input-data-types: component-identifier*s within *BasicRecord*:	"table1-a" "table1-b" "table1-c"
	*output-data-types: component-identifier*s within *BasicRecord*:	"y" "i" "q"
YIQColourSpace:	*y-band-identifier*:	"y"
	i-band-identifier:	"i"
	q-band-identifier:	"q"

AVA Documents Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996

2.) Let us assume, the input values are to be interpreted according to an RGB colour system ("r", "g", "b") where the "r", "g", and "b" bands take integer values in the range of (0..2), (0..4), and (0..1), respectively. The look-up table shown below maps the 24 possible RGB value triples onto YIQ colour values:

ImageStructure:	*band-identifier*s within *BandRecord*:	"r" "g" "b"
LookUpTable:	*number-of-input-dimensions*	3
	input-data-types	*BasicRecord* (3, ("r", non-negative integer), ("g", non-negative integer), ("b", non-negative integer))
	number-of-output-dimensions	3
	output-data-types	*BasicRecord* (3, ("Y", real), ("i", real), ("q", real))
	number-of-entries	24
	input-values	((0,0,0), (0,0,1), ..., (2,3,1))
	output-values	((.0,.0,.0), (.10,.20,.04), ..., (1.0,.0,.0))
YIQColourSpace:	*y-band-identifier*:	"y"
	i-band-identifier:	"i"
	q-band-identifier:	"q"

Syntax

```
LookUpTable ::= CHOICE
  {
  data-specified SEQUENCE
   {
   number-of-input-dimensions  [0] IMPLICIT INTEGER (1..MAX), -- No. 702
   input-data-types            [1] BasicDataType,             -- No. 602
   number-of-output-dimensions [2] IMPLICIT INTEGER (1..MAX), -- No. 702
   output-data-types           [3] BasicDataType,             -- No. 602
   number-of-entries           [4] IMPLICIT INTEGER (1..MAX), -- No. 702
   interpolation-method        [5] IMPLICIT CharacterString
                                  OPTIONAL,                   -- No. 722
   input-values                [6] IMPLICIT SEQUENCE OF
                                  BuiltinValue,               -- No. 206
   output-values               [7] IMPLICIT SEQUENCE OF
                                  BuiltinValue                -- No. 206
   },
  data-required                [8]  IMPLICIT NULL             -- No. 705
  }
```

Constraints

The value of the *number-of-input-dimensions* component shall equal the number of elementary data types which are declared by the *input-data-types* component.

The value of the *number-of-output-dimensions* component shall equal the number of elementary data types which are declared by the *output-data-types* component.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

Let n be the value of the *number-of-entries* component, d_{in} the value of the *number-of-input-dimensions* component, and d_{out} the value of the *number-of-output-dimensions* component. Then the number of *BuiltinValue* entities contained in the *input-values* component shall equal $n * d_{in}$ and the number of *BuiltinValue* entities contained in the *output-values* component shall equal $n * d_{out}$.

The data types of the *BuiltinValue* entities contained in the *input-values* sequence shall match the data type specification in the *input-data-types* component.

The data types of the *BuiltinValue* entities, contained in the *output-values* component, shall match the data type specification in the *output-data-types* component.

Choosing the *data-required* component outside the definition of segment type will cancel the specification of *LookUpTable* entity and may cause unpredictable results.

Documents
Document Sharing Hub
Full document details: ISO IEC 12087-3-1995 amd1-1996

IIF syntax entity No. 305 *RegionOfInterest*

Semantics

The *RegionOfInterest* entity stands for the description of a region of interest (ROI), as defined in ISO/IEC 12087-1 and used in ISO/IEC 12087-2. Five alternatives are provided for the representation of an ROI: a Boolean array, an ellipse, an n-dimensional interval, a polygon, and a set of coordinates.

a) The *array* component allows to represent the ROI by an array of Boolean values. This is the most general representation of an ROI because no constraints exist regarding its shape.

b) The *ellipse* component allows representation of the ROI by the geometrical shape of an ellipse.

c) The *rectangular* component allows representation of the ROI by an n-dimensional interval. This representation can be used to express rectangular regions.

d) The *polygon* component allows representation of the ROI by a sequence of coordinates that form a closed polygon. The region of interest is the inside of the polygon including its boundaries.

e) The *set-of-coordinates* component allows representation of the ROI by naming all coordinates that belong to the ROI. This representation is isomorphic to the Boolean array representation.

The *polarity-reversed* component may be used to define reversed polarity of the region of interest. If *polarity-reversed* is 'true', then the real ROI is the complementary set of the ROI actually described.

The *index-manipulation* component may be used in combination with the ROI types *ellipse*, *rectangular*, and *polygon* for the assignment of array dimensions as specified in ISO/IEC 12087-2.

The component is useful in the specification of a segment type. If this option is chosen in the specification of a segment type, an application can use, observing all existing constraints, any other component of *RegionOfInterest* entity at the time an IIF data stream is created or processed.

NOTE - The *array* and *set-of-coordinates* components provide an explicit description of the ROI whereas the other components provide generic descriptions.

Syntax

```
RegionOfInterest ::= CHOICE
   {
   data-specified SEQUENCE
       {
     roi-type CHOICE
        {
        array               [0] BooleanArray,                    -- No. 306
        ellipse             [1] Ellipse,                         -- No. 307
        rectangular         [2] IntervalND,                      -- No. 308
        polygon             [3] SetOfCoordinates,                -- No. 311
        set-of-coordinates  [4] SetOfCoordinates                 -- No. 311
        },
     polarity-reversed      [5] IMPLICIT BOOLEAN DEFAULT FALSE,-- No. 701
     index-manipulation     [6] IMPLICIT SEQUENCE OF
                                Identifier OPTIONAL,             -- No. 107
     },
   data-required            [7] IMPLICIT NULL                    -- No. 705
   }
```


AVA Documents Document Sharing Hub

Full document details: ISO IEC 12087-3-1995 amd1-1996