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)

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*.

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*).

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.

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.

*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*

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*)*

5.3.8 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*

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*

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

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.

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.

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.

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 attributc", dcfined 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, clementary 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.

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 *n*th 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 *n*th 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 "/".

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.

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.

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.

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.

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.

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.

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

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-identifiers* 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.

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.

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
   }
```

Constraints

The *index-manipulation* component may be used only in combination with the ROI types *ellipse*, *rectangular*, and *polygon*. The number of *Identifier* entities contained in the *index-manipulation* component shall equal the dimensionality of the ROI.

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

IIF syntax entity No. 315 ***FeatureList***

and

IIF syntax entity No. 316 ***CoordinateAndFeature***

Semantics

The *FeatureList* entity stands for the description of a feature list, as defined in ISO/IEC 12087-1.

The size of the list is given by the *number-of-coordinates* component, and the list of coordinates, together with the associated features, is given by the *coordinates-and-features* component.

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 *FeatureList* entity at the time an IIF data stream is created or processed.

The *CoordinateAndFeature* entity stands for the description of an n-tuple coordinate, given by the *coordinate* component, and of the associated feature, given by the *feature* component.

Syntax

```
FeatureList ::= CHOICE
 {
 data-specified SEQUENCE
   {
   number-of-coordinates      [0] IMPLICIT INTEGER (1..MAX),    -- No. 702
   coordinates-and-features   [1] IMPLICIT SEQUENCE OF
                                      CoordinateAndFeature,  -- No. 316
   },
   data-required              [3] IMPLICIT NULL                 -- No. 705
 }


CoordinateAndFeature ::= SEQUENCE
   {
   coordinate    [0] IMPLICIT CoordinateND,                     -- No. 310
   feature       [1] ANY
   }
```

Constraints

The number of *CoordinateAndFeature* entities contained in the *coordinates-and-features* component shall equal the value of the *number-of-coordinates* component.

The dimensionality of all *CoordinateND* entities contained in the *CoordinateAndFeature* entity shall equal the dimensionality of the image structure to which the region of interest is attached.

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

IIF syntax entity No. 401 *ImageAttribute*

Semantics

The *ImageAttribute* entity stands for the description of any of the image-related attributes defined in ISO/IEC 12087-1. For the rationale behind these attribute classes, refer to ISO/IEC 12087-1.

Instead of explicitly specifying one of these attributes, the *ImageAttribute* entity may also reference the desired attribute (via an image attribute reference) from the name space of the current segment or from the name space provided by an external address. The *freeform* components allows to connect to an application specific form of the definition of image attributes. These may be included in current IIF data stream or may reside outside it. The syntax of the *freeform* definition of attributes is not restricted to ASN.1.

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 *ImageAttribute* entity at the time an IIF data stream is created or processed

Syntax

```
ImageAttribute ::= CHOICE
    {
    metric              [0] MetricDescription,                  -- No. 402
    channel             [1] ChannelCharacteristi                -- No. 408
    colour              [2] ColourRepresentation,               -- No. 410
    control             [3] PIKSControl,                        -- No. 434
    freeform            [4] ANY,
    data-required       [5] IMPLICIT NULL                       -- No. 705
    }
```

Constraints

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

NOTE - Attributes may be attached not only to *FundamentalImageStructure* entities but also to higher-level structures. Thus, a single attribute may be declared for multiple images (e.g., by defining a record of images and attaching the attribute to the record). For this reason, the expression "... shall match all *FundamentalImageStructure* entities the attribute is declared for" is used within the list of constraints defined for the sub-entities of the *ImageAttribute* entity.

IIF syntax entity No. 402 *MetricDescription*

Semantics

The *MetricDescription* entity stands for the description of all metric attributes of an image, as defined in ISO/IEC 12087-1.

NOTE - The *MetricDescription* entity comprises information which belongs to the "representation attribute" class, defined in ISO/IEC 12087-1.

The *coordinate-system* component stands for the description of a reference coordinate system. The supported systems are Cartesian and angular. The orientation of the coordinate systems, with respect to an observer, is defined in ISO/IEC 12087-1. The following alternatives are predefined: *one-dimensional cartesian, two-dimensional cartesian, three-dimensional cartesian, four-dimensional cartesian, five-dimensional cartesian, other cartesian, two-dimensional polar, three-dimensional cylindrical, three-dimensional spherical,* and *other angular*. Additional definitions are subject to registration (see 6.2).

The *dimension-mappings* component allows expression of a relation between the indices of the dimensions and the coordinate system. The *metric-transform* component allows for the specification of a metric transformation of image data related to the coordinate system. If this component is absent, it is assumed that no transformation is applied to the image data. The *domain* component allows one to specify whether the image data are in the spatial or in the transform domain. If this component is absent, it is assumed that the image data are represented in the original space domain.

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 *MetricDescription* entity at the time an IIF data stream is created or processed.

Syntax

```
MetricDescription ::= CHOICE
 {
 data-specified SEQUENCE
   {
   coordinate-system  [0] IMPLICIT INTEGER {                    -- No. 702
                                    cartesian-1d(1),
                                    cartesian-2d(2),
                                    cartesian-3d(3),
                                    cartesian-4d(4),
                                    cartesian-5d(5),
                                    cartesian-other(6),
                                    polar-2d(7),
                                    cylindrical-3d(8),
                                    spherical-3d(9),
                                    angular-other(10)
                                    },
   dimension-mappings    [1] IMPLICIT SEQUENCE OF
                                    DimensionMapping,   -- No. 403
   metric-transform      [2] MetricTransformation OPTIONAL,    -- No. 406
   domain                [3] IMPLICIT Domain OPTIONAL          -- No. 407
   },
 data-required           [4] IMPLICIT NULL                     -- No. 705
 }
```

Constraints

The coordinate system shall match the dimensionality of the *FundamentalImageStructure* entities for which it is declared. This matching depends on the meaning of the dimensions and coordinate axis defined within the

DimensionMapping entity. Only values in the range of (1..10) and values which have been internationally registered are allowed for the *coordinate-system* component. Refer to 6.2.

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

IIF syntax entity No. 408 ***ChannelCharacteristics***

Semantics

The *ChannelCharacteristics* entity stands for the description of image capture-related attributes, as defined in ISO/IEC 12087-1.

The *band-identifier* component is used to associate the channel characteristics information with those bands of an image structure which have the same *band-identifier*. This component is optional. If it is absent, the channel characteristics description is associated with the whole image.

The *verbal-description* component provides a field for application-specific verbal descriptions of the channel characteristics.

The *precision* component specifies the number of decimal places that shall be regarded as significant. It may be used to describe the channel's acquisition accuracy.

The *channel-type* component may be used to express the type of channel that has been used to acquire or generate a channel (i.e., - in the wording of the data types defined in ISO/IEC 12087-1 - a band). The following alternatives are predefined: *bi-level, half-tone, grey-value, colour,* and *feature*. Additional definitions are subject to registration as defined in 6.2.

The *channel-usage* component may be used to express the usage of the channel with respect to the image's other channels (bands). The following alternatives are predefined: *opaque, transparent*. Additional definitions are subject to registration as defined in 6.2.

The *background-value* component may be used to express to what pixel value the image's background is assigned.

The *compandor-description* component allows to specify the compandor (compression/expander) model of the given channel.

The *sampling-function* component allows to specify the spatial sampling characteristics of the given channel.

The *application-specifics* component allows for application-specific extensions to the *ChannelCharacteristics* entity in general.

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 *ChannelCharacteristics* entity at the time an IIF data stream is created or processed.

Syntax

```
ChannelCharacteristics ::= CHOICE
 {
  data-specified SEQUENCE
  {
  band-identifier          [0] IMPLICIT Identifier       OPTIONAL,-- No. 107
  verbal-description       [1] IMPLICIT CharacterString OPTIONAL,-- No. 006
  precision                [2] IMPLICIT INTEGER          OPTIONAL,-- No. 702
  channel-type             [3] IMPLICIT INTEGER                   -- No. 702
                               {
                               bi-level(1),
                               half-tone(2),
                               grey-value(3),
                               colour(4),
                               feature(5)
                               } OPTIONAL,
  channel-usage            [4] IMPLICIT INTEGER                   -- No. 702
                               {
                               opaque(1),
                               transparent(2)
                               } OPTIONAL,
  background-value         [5] BuiltinValue              OPTIONAL,    -- No. 206
  compandor-description    [6] IMPLICIT
                               CompandorDescription      OPTIONAL,    -- No. 409
  sampling-function        [7] ANY                       OPTIONAL,
  application-specifics    [8] ANY                       OPTIONAL
  },
 data-required             [9] IMPLICIT NULL                      -- No. 705
}
```

Constraints

For the *channel-type* component, only values in the range of (1..5) and values which have been internationally registered are permitted. Refer to 6.2.

For the *channel-usage* component, only values in the range of (1..2) and values which have been internationally registered are permitted. Refer to 6.2.

For the *background-value* component, the type which is used to represent the value has to match the elementary pixel type of the pixel array to which this channel description is associated.

If a channel type is *colour*, the existence of a *ColourRepresentation* entity is required.

The encoded representation of a field of pixel values (*SingleDataUnit* entity) must provide at least as much bits per pixel as specified by the value of the *precision* component.

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

IIF syntax entity No. 410 *ColourRepresentation*

Semantics

The *ColourRepresentation* entity stands for the selection of a colour space including all attributes which are required to unambiguously describe the colour representation. According to the description in ISO/IEC 12087-1, two kinds of colour spaces are provided: standardized spaces that are based on the CIE-1931 colour system, given by the *standardized-space* component and non-standardized spaces, given by the *non-standardized-space* component.

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 *ColourRepresentation* entity at the time an IIF data stream is created or processed.

Syntax

```
ColourRepresentation ::= CHOICE
 {
 data-specified SEQUENCE
   {
   colour-space-type CHOICE
      {
      standardized-space        [0] StandardizedSpace,          -- No. 411
      non-standardized-space    [1] NonStandardizedSpace        -- No. 424
      },
   test-colour     [2] IMPLICIT TestColour OPTIONAL             -- No. 433
   },
 data-required     [3] IMPLICIT NULL                            -- No. 705
}
```

Constraints

See constraints of the subentities.

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

IIF syntax entity No. 501 *ImageAnnotation*

Semantics

The *ImageAnnotation* entity stands for the description of text, graphics, audio, or other application-related data. The annotation may be semantically attached to an image as a whole or to a specific location within the image using the *location* component.

The *numeric-identifier* component provides one of two ways to specify how the *contents* component has to be interpreted. The following alternatives are predefined:

"1": *plain text contents*

"2": *structured text contents* according to ISO/IEC 8613, Open Document Architecture (ODA), Office Document Format (FOD) 11 (simple text documents), ISO/IEC 10000.

"3": *structured text contents* according to ISO/IEC 8879, Standard Generalized Markup Language (SGML), represented according to ISO/IEC 9069, SGML Document Interchange Format (SDIF).

"4": *geometric graphics contents* according to ISO/IEC 8632, Computer Graphics Metafile (CGM).

"5": *geometric graphics contents* according to ISO/IEC 8632, Computer Graphics Metafile (CGM), Revision 1992.

"6": *audio contents* according to CCITT Recommendation G.711, encoding type *aLaw*.

"7": *audio contents* according to CCITT Recommendation G.711, encoding type *muLaw*.

"8": *audio contents* according to CCITT Recommendation G.721.

"9": *audio contents* according to ISO/IEC 11172 (MPEG-1).

Additional definitions are subject to registration as defined in 6.2.

As an alternative, the *object-identifier* component may be used to specifies how the *contents* component has to be interpreted by identifying an officially registered object.

The contents of the annotation is either directly represented by the *contents* component, or via a reference to an image data unit, using the *reference-to-image-data* component. In the former case, the syntax of the *contents* component depends on the contents type as specified by either the *numeric-identifier* component or the *object-identifier* component. The latter case may be used to refer to an audio data stream that is stored interleaved with image data within a *ReferencedUnit* entity, e.g., using the MPEG-1 encoding.

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 *ImageAnnotation* entity at the time an IIF data stream is created or processed.

NOTE - The semantics contained in an image annotation are only meaningful to certain applications. No meaning is defined throughout ISO/IEC 12087. However, it is assumed that the application-specific semantics contained in a certain image annotation are related to all substructures with which they are associated. If *ImageAnnotation* entities occur, the IIF parser shall switch to the text, graphics, audio, or application-specific processor as indicated by either the *numeric-identifier* or *object-identifier* component.

Syntax

```
ImageAnnotation ::= CHOICE
 {
 data-specified SEQUENCE
   {
   location                      [0] Location OPTIONAL,          -- No. 502
   contents-descriptor CHOICE
      {
      numeric-identifier         [1] IMPLICIT INTEGER            -- No. 702
                                     {
                                     plain-text(1),
                                     oda-fod-11(2),
                                     sgml(3),
                                     cgm-87(4),
                                     cgm-92(5),
                                     g711-a-law(6),
                                     g711-mu-law(7),
                                     g721(8),
                                     mpeg-1(9)
                                     },
      object-identifier          [2] IMPLICIT OBJECT IDENTIFIER -- No. 706
      },
   contents-type CHOICE
      {
      contents                   [3] ANY,
      reference-to-image-data    [4] IMPLICIT Identifier        -- No. 107
      }
   },
 data-required                  [5] IMPLICIT NULL               -- No. 705
 }
```

Constraints

For the *numeric-identifier* component, only values in the range of (1..9) and values which have been internationally registered are permitted. Refer to 6.2.

For contents of type *plain text*, the syntax of the *contents* component shall be CharacterString. For the other content types, the syntax depends on the encoded representation provided by the referenced standards:

a) clear text encoded content types shall be represented by CharacterString

b) binary encoded content types shall be represented by OCTET STRING

c) content types whose representations are given by an ASN.1 syntax shall be represented according to this syntax.

The *reference-to-image-data* component may only be chosen in combination with a *ReferencedUnit* entity which contains interleaved data in a well-defined representation.

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

NOTE - The only well-defined representation for image and audio data defined by this Standard is MPEG-1. Future representations are subject to registration. Refer to 6.2.

IIF syntax entity No. 602 *BasicDataType*

and

IIF syntax entity No. 603 *CompoundDataType*

Semantics

The *BasicDataType* entity stands for the description of a basic data type, as defined in ISO/IEC 12087-1. The components are called *compound-data-type* and *elementary-data-type*, respectively.

Instead of explicitly specifying one of these basic data type components, the *BasicDataType* entity may also reference the desired basic data type (via a basic data type reference) from the name space of the current segment or from the name space provided by an external address.

The *CompoundDataType* entity covers the description of a compound data type, as defined in ISO/IEC 12087-1. It can be an array of basic data types, called a *basic-array*, a record of basic data types, called a *basic-record*, a list of basic data types, called a *basic-list*, or a set of basic data types, called *basic-set*.

The *data-required* component is usable in the specification of a segment type. If this option is chosen in the definition of a segment type, the value of *BasicDataType* entity remains unspecified. An application can then, under observation of all existing constraints, decide at the time an IIF data stream is created or processed, what kind of data type is the best in given segment type for image interchange.

Syntax

```
BasicDataType ::= CHOICE
    {
    compound-data-type      [0] CompoundDataType,              -- No. 603
    elementary-data-type    [1] ElementaryDataType,            -- No. 609
    referenced-data-type    [2] BasicDataTypeRef,              -- No. 930
    data-required           [3] IMPLICIT NULL                  -- No. 705
    }


CompoundDataType ::= CHOICE
    {
    basic-array         [0] BasicArray,                        -- No. 604
    basic-record        [1] BasicRecord,                       -- No. 605
    basic-list          [2] BasicList,                         -- No. 607
    basic-set           [3] BasicSet,                          -- No. 608
    data-required       [4] IMPLICIT NULL                      -- No. 705
}
```

Constraints

Choosing the *data-required* component outside the definition of segment type will cancel the specification of *BasicDataType* entity and may cause unpredictable results when accessing or processing image data.

IIF syntax entity No. 609 *ElementaryDataType*

Semantics

The *ElementaryDataType* entity stands for the description of an elementary data type, as defined in ISO/IEC 12087-1. The following alternatives are provided: *boolean, non-negative integer, signed integer, real, complex,* and *enumeration.*

Syntax

```
ElementaryDataType ::= CHOICE
    {
    elementary-data-type INTEGER                                    -- No. 702
                           {
                           boolean(1),
                           non-negative-integer(2),
                           signed-integer(3),
                           real(4),
                           complex(5),
                           enumeration(6)
                           },
    data-required     [1] IMPLICIT NULL                             -- No. 705
    }
```

Constraints

For the *ElementaryDataType* entity, only values in the range of (1..6) and values which have been internationally registered are permitted. Refer to 6.2.

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

Add the following new subclause:

5.3.8 Entities for the description of segmentation facility

IIF syntax entity No. 901 *SegmentProlog*

Semantics

The *SegmentProlog* entity stands for the definition of 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. 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 *segment-id* component is a label, which is used to reference this segment from other segments. References to labelled segments are not limited to nested segments; they can be referenced from any segment in the IIF data stream, or from outside.

The set of *user-label* components is an additional information about the segment; it can be freely chosen and used by an application e.g. to filter the segment with associated set of names (semantic tags) from the whole IIF data stream. An ASN.1 I/O subsystem can optionally support the corresponding functunality (called commonly "name sets"). No further semantics is given on the *user-label* component.

NOTE - Usually a *segment-id* is assigned by the IIF Gateway while *user-label* may be given by an application. Therefore any provision on information interchange based on *user-label* component is application, or application class specific.

The *segment-type* component specifies the required segment structure and the default set of attributes.

The *segment-attributes* component models generic information necessary for understanding of the segment body by an application or by the user of an application.

The *named-items* component provides type definitions which can be used in the name space of the current segment or referenced by other segments.

Syntax

```
SegmentProlog ::= SEQUENCE {
  segment-id          [0] IMPLICIT SegmentLabel       OPTIONAL, -- No. 923
  user-label          [1] IMPLICIT SEQUENCE OF Label  OPTIONAL, -- No. 925
  segment-type        [2] IMPLICIT SegmentTypeRef     OPTIONAL, -- No. 931
  segment-attributes  [3] IMPLICIT SegmentAttributes  OPTIONAL, -- No. 903
  named-items         [4] IMPLICIT NamedItems         OPTIONAL  -- No. 904
}
```

Constraints

The label used to reference the segment must be unique within an IIF data stream The user label is not constrained in any way.

IIF syntax entity No. 902 *SegmentEpilog*

Semantics

The *SegmentEpilog* entity stands for the definition of the boundary of a segment within an IIF data stream, and as a container for information which may facilitate random access to an IIF data stream residing in the memory buffer or on a file.

The *end-markup* component is an empty construct which denotes the end of a segment if this is required in specific context.

The *access-optimizer* component provides additional (application specific) information which will facilitate random access to the content of the current IIF data stream.

EXAMPLE - The *access-optimizer* component can provide absolute byte offsets to the nodes of its sub-structure, i.e. nested segments, and backward pointer to its parent node.

Syntax

```
SegmentEpilog ::= CHOICE {
  end-markup              [0] IMPLICIT NULL,                    -- No. 705
  access-optimizer        [1] EXTERNAL                          -- No. 708
}
```

Constraints

None

IIF syntax entity No. 903 *SegmentAttributes*

Semantics

The *SegmentAttributes* entity stands for a definition of any of *image-related data*, *image-attributes*, *image-annotations*, and *basic-data* components as defined in IPI-CAI. The usage of attributes in a given segment can be constrained by the definition of a segment type. In contrast to corresponding components of the *ContentsBody* entity, values of components of this entity apply to elements of the current segment and can be inherited in its nested segments. The inheritance mechanism is controlled by the segment type definition.

Syntax

```
SegmentAttributes ::= SEQUENCE {
  image-related-data    [0] IMPLICIT SEQUENCE OF
                            ImageRelatedData OPTIONAL,    -- No. 301
  image-attributes      [1] IMPLICIT SEQUENCE OF
                            ImageAttribute   OPTIONAL,    -- No. 401
  image-annotation      [2] IMPLICIT SEQUENCE OF
                            ImageAnnotation  OPTIONAL,    -- No. 501
  basic-data-object     [3] IMPLICIT SEQUENCE OF
                            BasicDataObject  OPTIONAL     -- No. 602
}
```

Constraints

None.

Add the following new subclause:

5.3.9 Entities for the description of segment types

IIF syntax entity No. 904 *NamedItems*

Semantics

The *NamedItems* entity provides the crucial part of referencing mechanism. It is used to specify any of named image structures, image related data, image attributes, image annotations, basic data types and segment types. In other words, this syntax entity must be present in the prolog of a segment if some of its components are the targets for a reference. Their components can be referenced by name within the name space of the current segment unless an optional external address points to a different name space.

Syntax

```
NamedItems ::= SEQUENCE {
  image-structure-defn        [0] IMPLICIT SEQUENCE OF
                                  ImageStructureDefn    OPTIONAL, -- No. 905
  image-related-data-defn     [1] IMPLICIT SEQUENCE OF
                                  ImageRelatedDataDefn  OPTIONAL, -- No. 906
  image-attributes-defn       [2] IMPLICIT SEQUENCE OF
                                  ImageAttributeDefn    OPTIONAL, -- No. 907
  image-annotation-defn       [3] IMPLICIT SEQUENCE OF
                                  ImageAnnotationDefn   OPTIONAL, -- No. 908
  basic-data-types-defn       [4] IMPLICIT SEQUENCE OF
                                  BasicDataTypeDefn     OPTIONAL, -- No. 909
  segment-type-defn           [5] IMPLICIT SEQUENCE OF
                                  SegmentTypeDefn       OPTIONAL  -- No. 910
}
```

Constraints

None.

IIF syntax entity No. 905 *ImageStructureDefn*

and

IIF syntax entity No. 906 *ImageRelatedDataDefn*

and

IIF syntax entity No. 907 *ImageAttributeDefn*

and

IIF syntax entity No. 908 *ImageAnnotationDefn*

and

IIF syntax entity No. 909 *BasicDataObjectDefn*

Semantics

The components of the *NamedItems* entity are modelled by the *ImageStructureDefn*, *ImageRelatedDataDefn*, *ImageAttributeDefn*, *ImageAnnotationDefn*, and *BasicDataTypeDefn* entities. These provide a mechanism to associate a name with some objects defined in the IPI-CAI. The names are modelled as labels of different types.

Syntax

```
ImageStructureDefn ::= SEQUENCE {
  image-type-id    [0] IMPLICIT ImageStructureLabel,          -- No. 918
  image-structure  [1] IMPLICIT ImageStructure                -- No. 102
}

ImageRelatedDataDefn ::= SEQUENCE {
  image-related-data-id [0] IMPLICIT ImageRelatedDataLabel,   -- No. 919
  image-related-data    [1] IMPLICIT ImageRelatedData         -- No. 301
}

ImageAttributeDefn ::= SEQUENCE {
  image-attribute-id [0] IMPLICIT ImageAttributeLabel,        -- No. 920
  image-attribute    [1] IMPLICIT ImageAttribute              -- No. 401
}

ImageAnnotationDefn ::= SEQUENCE {
  image-annotation-id [0] IMPLICIT ImageAnnotationLabel,      -- No. 921
  image-annotation    [1] IMPLICIT ImageAnnotation            -- No. 501
}

BasicDataObjectDefn ::= SEQUENCE {
  data-type-id        [0] IMPLICIT BasicDataTypeLabel         -- No. 922
  basic-data-object   [1] IMPLICIT BasicDataType              -- No. 602
}
```


Constraints

None.

IIF syntax entity No. 910 *SegmentTypeDefn*

Semantics

The *SegmentTypeDefn* entity models the required structure of a segment, its required set of attributes and their usage, and the segment specific image content. It allows also to associate with a given segment type specific format constraints, and a specific set of common information.

The *segment-type-ID* component is the unique symbolic name of the type being defined. It can be referenced by this name within the name space of the current segment unless an optional external address points to a different name space. The *segment-type-label* component is an additional information about the segment type, it can be freely chosen and used by an application e.g. to filter the segment definition with associated names (semantic tags) from the whole IIF data stream. No further semantics is given on *segment-type-label.*

The *segment-structure* component describes a set of constraints on the order, grouping, and number of previously defined types and how to bind attributes to these.

NOTE - The segment is of a predefined type if the topology of its body satisfies the topology described in the type definition and all other features specified there can be observed.

The *segment-format* component contains the same information as the *FormatDescriptor* entity does for the IIF data stream. These definitions are, however, restricted in scope to a segment of given type and they obscure any other information of the same type available by inheritance or indirect reference.

The *segment-descriptor* component contains the same information as the *ContentsHeader* entity does for the IIF data stream. These definitions are, however, restricted in scope to a segment of given type and they obscure any other information of the same type available by inheritance or indirect reference.

NOTE - Both *segment-format* and *segment-descriptor* components facilitate the integration of other data in an IIF data stream. This data can follow foreign syntactic rules. For instance, an IIF data stream can be produced according to another version of IIF grammar, or an IIF data stream can have a different copy-right owner.

NOTE - *Segment-format* and *segment-descriptor* components, while present in a segment type definition deep within the body of an IIF data stream, may cause serious processing problems. In order to solve these problems, critical information entities should be elevated as far as possible towards the beginning of the current IIF data stream (to *format-descriptor* and *content-header* components of the *FullDataFormat* entity).

Syntax

```
SegmentTypeDefn ::= SEQUENCE {
  segment-type-ID    [0] IMPLICIT SegmentTypeLabel,              -- No. 924
  segment-type-label [1] IMPLICIT SEQUENCE OF Label OPTIONAL,    -- No. 925
  segment-structure  [2] IMPLICIT SEQUENCE OF
                                  ExpressionElement OPTIONAL,    -- No. 911
  segment-format     [3] IMPLICIT FormatDescriptor  OPTIONAL,    -- No. 002
  segment-descriptor [4] IMPLICIT ContentsHeader    OPTIONAL     -- No. 005
}
```

Constraints

The name of the segment type must be unique within an IIF data stream. The *segment-type-label* is not constrained in any way.

IIF syntax entity No. 911 *ExpressionElement*

Semantics

The *ExpressionElement* entity defines the context and structure of current segment. The context is specified by a set of required content attributes and their presence and propagation rules in the *attribute-occurrence-list* component, and by the so-called specific image. The structure of the segment, i.e. the topology of nested segments, is specified by recursive invocation of the *StructureDefn* entity in the *structure-element* component. The specific image is an image content bound to *prolog* of some segment rather than occurring in its content. It provides an application with compulsory context which makes it possible to interpret the content of the current segment and its nested segments in a correct way. The *specific-image-structure* component specify how a specific image is structured while the *specific-reference-unit* component specifies its data.

The optional *logical-address* component specify the address saying where a given substructure shall be located within parent structure. The *node-ID* component defines an unique address within the current structure. This address be used by the *logical-address* construct.

NOTE - Segment type with specific content can be used to define constant image, image related, or image annotation contents which is useful in further type definitions. In such a way they will facilitate fulfilling of special e.g. legal requirements like the presence of a company logo or copyright information.

Syntax

```
ExpressionElement ::= SEQUENCE {
  node-ID                    [0] IMPLICIT SegmentTypeLabel,       -- No. 924
  attribute-occurrence-list  [1] SEQUENCE OF
                                 AttributeOccurrence OPTIONAL,    -- No. 912
  specific-image-structure   [2] ImageStructure   OPTIONAL,       -- No. 102
  specific-reference-unit    [3] ReferenceUnit    OPTIONAL,       -- No. 201
  structure-element          [4] StructureDefn    OPTIONAL,       -- No. 913
  logical-address            [5] SegmentTypeLabel OPTIONAL        -- No. 924
}
```

Constraints

Any modification of, or any addition to the specific content would potentially change the semantics associated with the segment type and therefore is not allowed without changing the segment type.

The name of an ExpressionElement must be distinctive in a given context. Therefore the value of name-ID must be unique within an IIF data stream.

IIF syntax entity No. 912 *AttributeOccurrence*

Semantics

The *AttributeOccurrence* entity describes the rule of presence and propagation for an attribute defined in the specific node of a structure given by the segment type definition. This rule is a combination of predicates (called also meta-attributes, i.e. attributes of attributes). Predicates shall be observed if the concerned attribute is used by an application to process or to create the content of a segment of given type.

The *attribute-type* component specifies to which kind of attribute the presence and propagation rule will be applied. The specification of a selected attribute type applies to the content of the current segment, and its nested segments, unless explicit predicates suspend the inheritance.

The *attribute-occurrence-type* component models the predicates in the form of a bit string for each single specified attribute. A set bit means that the corresponding predicate must be adopted to the selected attribute.

If the *public* bit is set, the value of an attribute can be specified in the segment type definition. The value of an attribute may or may not be specified by an application. If a value of an attribute is set, it applies to the content of the current segment, and to its nested segments, unless another predicate suspends the inheritance. The value of a *public* attribute can be changed at any time by an application.

If the *required* bit is set, the value of an attribute can be specified in the segment type definition. It is then default for the current segment and its nested segments unless another predicate suspends the inheritance. The default value of a *required* attribute can be changed by an application and redefined along an inheritance path individually for each nested segment. If the value of a *required* attribute is not set in the segment type definition, it shall be provided by an application for the current segment. It will be then inherited in nested segments if the *current* predicate is also specified, unless an application shall provide that value for each nested segment.

If the *fixed* bit is set, the value of an attribute shall be specified in the segment type definition. The type of a *fixed* attribute and its value applies to the content of the current segment and to its nested segments unless another predicate suspends the inheritance. The value of a *fixed* attribute can not be changed by an application, nor redefined along an inheritance path.

If the *implied* bit is set, the value of an attribute shall not be set in the segment type definition. The value shall be set by an application for the current segment and its nested segments depending on the contents of the specific segment.

If the *current* bit is set, the value of the attribute of the parent node is inherited even if it is different from the default value of the attribute specified by the segment type definition. The value of a *current* attribute can be changed at any time by an application.

If the *local* bit is set, the type and value of an attribute applies to the current segment only, and it is not inherited by its nested segments.

If the *extend* bit is set, the child segment completes the specification of a type of an attribute provided in the parent segment, and it can extend the value of an attribute inherited from the parent segment, respectively.

If the *obscure* bit is set, the child segment overwrites the type and value of the same attribute specified in the parent segment.

If the *stop* bit is set, the attribute inheritance for the specified attribute terminates on the current node the of segment structure (i.e. it should be considered as not specified for a child segment).

Syntax

```
AttributeOccurrence ::= SEQUENCE {
  attribute-occurrence-type  [0] IMPLICIT BIT STRING  {          -- No. 703
    public    (0),
    required  (1),
    fixed     (2),
    implied   (3),
    current   (4),
    local     (5),
    extend    (6),
    obscure   (7),
    stop      (8)
  } DEFAULT {public, current},

  attribute-type CHOICE {
    image-related-data     [1] ImageRelatedData,                 -- No. 301
    image-attributes       [2] ImageAttribute,                   -- No. 401
    image-annotation       [3] ImageAnnotation,                  -- No. 501
    basic-data-type        [4] BasicDataType                     -- No. 602
  }

}
```

Constraints

The predicates *public, fixed, required, implied* are mutually exclusive. The predicate *stop* is exclusive to all other predicates. The predicate *extend* and *obscure* are mutually exclusive.

Mutually exclusive predicates may not occur simultaneously in the *attribute-occurrence-type* component.

IIF syntax entity No. 913 *StructureDefn*

and

IIF syntax entity No. 914 *OccurrenceDefn*

Semantics

The *StructureDefn* entity models the structure of a set of segments that reference segment types. It describes a set of constraints on the order, grouping, and number of segments with type references. The *sequence-structure* component stands for a sequence of element occurrences that are constrained to occur in the specified order. The *set-structure* component stands for a set of element occurrences that are not constrained with respect to order. The *choice-structure* denotes a group of element occurrences from which only one may be chosen.

The *OccurrenceDefn* entity describes the number of times the element of a structure definition may occur, and whether it may be omitted. The *required-element* component means that the construction must occur as often as specified by *repeat-element* entity.The *optional-element* component means that the construction may occur as often as specified by *repeat-element* entity or not at all.

Syntax

```
StructureDefn ::= CHOICE {
  sequence-structure [0] IMPLICIT SEQUENCE OF OccurrenceDefn,    -- No. 914
  set-structure      [1] IMPLICIT SEQUENCE OF OccurrenceDefn,    -- No. 914
  choice-structure   [2] IMPLICIT SEQUENCE OF OccurrenceDefn     -- No. 914
}

OccurrenceDefn ::= CHOICE {
  required-element   [0] RepeatElement,                          -- No. 915
  optional-element   [1] RepeatElement                           -- No. 915
}
```

Constraints

None.

IIF syntax entity No. 915 *RepeatElement*

and

IIF syntax entity No. 916 *StructureElement*

Semantics

The *RepeatElement* defines the number of occurences of a structure element. The *min-repetition* and *max-repetition* components specify minimum and maximum values respectively. If *max-repetition* component is ommited the upper bound is unlimited.

The *StructureElement* entity is either a reference to a segment type, or it introduces the definition of a nested structure. The reference to a segment type provided by the choice of *reference-type* component in *StructureElement* entity terminates the recursive definition of a segment type.

Syntax

```
RepeatElement ::= SEQUENCE {
  structure-element [0] IMPLICIT StructureElement,          -- No. 916
  min-repetition    [1] IMPLICIT INTEGER DEFAULT 1,         -- No. 702
  max-repetition    [2] IMPLICIT INTEGER OPTIONAL           -- No. 702
}

StructureElement ::= CHOICE {
  referenced-type    [0] SegmentTypeRef,                    -- No. 931
  expression-element [1] ExpressionElement                  -- No. 911
}
```

Constraints

The value of *min-repetition* component must be greater or equal to one and less or equal to the value of *max-repetition* component.

The *referenced-type* component of the *StructureElement* entity shall not refer back the segment type inside which it is defined (prohibition of direct self-referencing) nor it is allowed to refer back to the embedding segment type in an indirect way, i.e., via some other segment type(s) defined in terms of this segment type (prohibition of indirect self-referencing).

Add the following new subclause:

5.3.10 Entities for the description of reference mechanisms

IIF syntax entity No. 917 *EntityHandle*

Semantics

The *EntityHandle* entity allows an application to constrain the scope of external data referred by the *ExternalReference* mechanism. The *segment-handle* defines an unique user readable name of an information entity predefined by the structural properties of the referenced data. Within an IIF data stream this name has well defined semantics (see *SegmentProlog* entity). If the referenced data stream does not follow the IIF grammar, the name specified as *segment-handle* shall be mapped by the application responsible for the handling of the referenced data to the form suitable for identification of needed information quanta.

The *in-file-handle* component allows to specify the data in an external repository in application specific way.

The *data-required* component is usable in the specification of a segment type. If this option is chosen in the definition of a segment type, the values of the *object-address* component and the *object-format* component of the *ExternalReference* entity remains unspecified and it should be provided, under observation of all existing constraints, by an application at the time an IIF data stream is created or processed.

Syntax

```
EntityHandle ::= CHOICE {
  segment-handle       [0] SegmentLabel,                    -- No. 923
  in-file-handle       [1] EXTERNAL,                        -- No. 708
  data-required        [2] IMPLICIT NULL                    -- No. 705
}
```

Constraints

Choosing the *data-required* component outside the definition of segment type will cancel the definition of *entity-address* component and may produce undesired results.

IIF syntax entity No. 918 ***ImageStructureLabel***

and

IIF syntax entity No. 919 ***ImageRelatedDataLabel***

and

IIF syntax entity No. 920 ***ImageAttributeLabel***

and

IIF syntax entity No. 921 ***ImageAnnotationLabel***

and

IIF syntax entity No. 922 ***BasicDataTypeLabel***

and

IIF syntax entity No. 923 ***SegmentLabel***

and

IIF syntax entity No. 924 ***SegmentTypeLabel***

and

IIF syntax entity No. 925 ***Label***

Semantics

Labels are used to identify and reference various entities of an IIF data stream. All label types are defined as *Label*, which in its turn is defined as *IA5String*. The comparison of labels for equality is case sensitive. For example, the labels "ABC" and "abc" are considered to be different labels.

Syntax

```
ImageStructureLabel     ::= Label                    -- No. 925

ImageRelatedDataLabel   ::= Label                    -- No. 925

ImageAttributeLabel     ::= Label                    -- No. 925

ImageAnnotationLabel    ::= Label                    -- No. 925

BasicDataTypeLabel      ::= Label                    -- No. 925

SegmentLabel            ::= Label                    -- No. 925

SegmentTypeLabel        ::= Label                    -- No. 925

Label ::= IA5String                                  -- No. 722
```

Constraints

In a given name space a *label* implies the data type to which it refers. Therefore only all labels of a given type must be unique. Strings modelling *Label* need not be unique across types, however. For example, it is legal (but

not encouraged!) to have the same label for both *ImageStructureLabel* and *ImageAnnotationLabel* within coinciding or identical scopes.

IIF syntax entity No. 926 *ImageStructureRef*

and

IIF syntax entity No. 927 *ImageRelatedDataRef*

and

IIF syntax entity No. 928 *ImageAttributeRef*

and

IIF syntax entity No. 929 *ImageAnnotationRef*

and

IIF syntax entity No. 930 *BasicDataTypeRef*

and

IIF syntax entity No. 931 *SegmentTypeRef*

Semantics

The following entities are required to reference named image structures, image related data, image attributes, image annotations, basic data types, and segment types, that modelled by a set of specific entities. These consist of an address to the IIF segment, or to some other defined repository where an appropriate definition is provided (the *external-address* component), and of the name of the facility (the *label* component) unique within the name space of an external segment or some other repository.

Syntax

```
ImageStructureRef ::= SEQUENCE {
  external-address [0] IMPLICIT SegmentRef OPTIONAL,          -- No. 932
  label            [1] IMPLICIT ImageStructureLabel           -- No. 929
}

ImageRelatedDataRef ::= SEQUENCE {
  external-address [0] IMPLICIT SegmentRef OPTIONAL,          -- No. 932
  label            [1] IMPLICIT ImageRelatedDataLabel         -- No. 919
}

ImageAttributeRef ::= SEQUENCE {
  external-address [0] IMPLICIT SegmentRef OPTIONAL,          -- No. 932
  label            [1] IMPLICIT ImageAttributeLabel           -- No. 920
}

ImageAnnotationRef ::= SEQUENCE {
  external-address [0] IMPLICIT SegmentRef OPTIONAL,          -- No. 932
  label            [1] IMPLICIT ImageAnnotationLabel          -- No. 921
}

BasicDataTypeRef ::= SEQUENCE {
  external-address [0] IMPLICIT SegmentRef OPTIONAL,          -- No. 932
  label            [1] IMPLICIT BasicDataTypeLabel            -- No. 922
}

SegmentTypeRef ::= SEQUENCE {
  external-address [0] IMPLICIT SegmentRef OPTIONAL,          -- No. 932
  label            [1] IMPLICIT SegmentTypeLabel              -- No. 924
}
```

Constraints

None.

IIF syntax entity No. 932 *SegmentRef*

and

IIF syntax entity No. 933 *ExternalRefIndex*

Semantics

The *SegmentRef* entity defines an address of the segment in the current or in a remote IIF data stream, or for an address of the quanta of information which can be distinguished by name within some structured remote data set. The semantics of the *SegmentRef* entity provides the most common part of functionality defined by the semantics of *ExternalAddress* entity (see there). The *segment-handle* component provides a name which must be unique in a given context. Within an IIF data stream this name has well defined semantics (see *SegmentProlog* entity). If the referenced data stream does not follow IIF grammar, the name specified as *segment-handle* shall be mapped by the application responsible for handling of referenced data to the form suitable for identification of needed information quanta. The remote data set is specified as *entity-index* component by an index into the list of *external-references* in the header of the current IIF data stream. The first item in this list is defined to have an index value of one.

Syntax

```
SegmentRef ::=  SEQUENCE {
  entity-index          [0] IMPLICIT ExternalRefIndex OPTIONAL,    -- No. 933
  segment-handle        [1] IMPLICIT SegmentLabel                  -- No. 923
}

ExternalRefIndex ::= INTEGER                                       -- No. 702
```

Constraints

In order to allow for a correct interpretation of external addresses for each given *entity-index* number, an appropriate entry must exist in the list of *external-references* in the header of the current IIF data stream.

IIF syntax entity No. 934 *ImageDataRef*

Semantics

The ImageDataRef entity defines an optional way to refer to the unit of pixel values that is modelled with a label to indicate to which it belongs. The full semantics of an access mechanism is provided by the description of *ReferenceUnit* entity.

Syntax

```
ImageDataRef ::= SEQUENCE {
  referenced-segment   [0] IMPLICIT SegmentRef OPTIONAL,          -- No. 932
  CHOICE {
    reference-by-name  [1] IMPLICIT Identifier,                   -- No. 107
    reference-by-index [2] IMPLICIT INTEGER                       -- No. 702
  }
}
```

Constraints

None.

In Annex A replace table 7 by the following:

IIF syntax entity		Component names	No.	Type	
001	FullDataFormat	format-descriptor	00 100	C	
		contents-header	00 101	C	
		contents	00 102	C	
002	FormatDescriptor	self-identification	00 200	E	
		version	00 201	C	
		profile	00 202	C	
003	Version	standard	00 300	E	
		publication-date	00 301	E	
004	Profile	application-profile	00 401	E	
		profile-definition	00 402		
005	ContentsHeader	title	00 500	C	
		owner	00 501	C	
		date-and-time	00 502	E	
		message	00 503	C	
		application-data	00 504	A	
		external-references	00 505	C	SEQ OF
		access-information	00 506	C	
		type-guide	00 507	C	
006	CharacterString	standard-characters	00 600	E	
		special-characters	00 601	C	
007	SpecialCharacterString	character-set-escape	00 700	E	
		characters	00 701	E	
008	Contents	--	--	C	SEQ OF
009	ContentsElement	prolog	00 900	C	
		body	00 901	C	SEQ OF
		epilog	00 902	C	
010	ContentsBody	image	01 000	C	
		image-related-data	01 001	C	
		image-attribute	01 002	C	
		image-annotation	01 003	C	
		basic-data-object	01 004	C	
		contents-element	01 005	C	

Table 7 - List of IIF-DF syntax entities and components

IIF syntax entity		Component names	No.	Type	
101	Image	image-structure	10 100	C	
		image-data	10 101	C	SEQ OF
102	ImageStructure	structure-type	10 200	C	HOICE
		compound-structure	10 201	C	
		fundamental-structure	10 202	C	
		image-attributes	10 203	C	SEQ OF
		image-related-data	01 001	C	SEQ OF
		image-annotations	10 205	C	SEQ OF
		processing-history	10 206	C	
		referenced-structure	10 207	C	
103	CompoundImageStructure	compound-image-array	10 300	C	
		compound-image-record	10 301	C	
		compound-image-list	10 302	C	
		compound-image-set	10 303	C	
104	CompoundImageArray	dimensionality	10 400	C	
		serialization	10 401	C	
		data-placement	10 402	C	
		element-structure	10 403	C	
105	Dimensionality	number-of-dimensions	10 500	E	
		dimension-descriptions	10 501	C	SEQ OF
106	DimensionDescription	dimension-identifier	10 600	C	
		lower-boundary	10 601	E	
		upper-boundary	10 602	E	
107	Identifier	--	--	E	
108	Serialization	--	--	C	SEQ OF
109	DataPlacement	--	--	E	
110	CompoundImageRecord	number-of-components	11 000	E	
		data-placement	10 402	C	
		record-components	11 002	C	SEQ OF
111	RecordComponent	component-identifier	11 100	C	
		component-structure	11 101	C	
112	CompoundImageList	number-of-elements	11 200	E	
		data-placement	10 402	C	
		element-structure	10 403	C	

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.	Type	
113	CompoundImageSet	number-of-members	11 300	E	
		data-placement	10 402	C	
		member-structure	11 302	C	
114	FundamentalImageStructure	band-record	11 400	C	
		metric-array	11 401	C	
115	BandRecord	number-of-bands	11 500	E	
		data-placement	10 402	C	
		record-components	11 002	C	SEQ OF
116	BandRecordComponent	band-identifier	11 600	C	
		component-structure	11 101	C	
117	MetricArray	dimensionality	10 400	C	
		serialization	10 401	C	
		data-placement	10 402	C	
		element-structure	10 403	C	
118	MetricArrayElement	pixel-structure	11 800	C	
		band-record	11 400	C	
		metric-array	11 401	C	
119	PixelStructure	elementary-pixel-structure	11 900	C	
		compound-pixel-structure	11 901	C	
120	ElementaryPixelStructure	--	--	E	
121	PixelBandRecord	number-of-components	11 000	E	
		record-components	11 002	C	SEQ OF
122	PixelBandRecordComponent	band-identifier	11 600	C	
		component-structure	11 101	C	

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.	Type	
201	ReferencedUnit	reference-label	20 100	C	
		explicit-values	20 101	C	
		referenced-values	20 102	C	
202	DataUnit	single-data-unit	20 200	C	
		subdivided-data-unit	20 201	C	
		external-data-unit	20 202	C	
		external-ref-index	20 203	C	
203	SubdividedDataUnit	number-of-pixels	20 300	E	
		partitions	20 301	C	SEQ OF
204	SingleDataUnit	builtin-encoded-data-unit	20 400	C	
		externally-defined-data-unit	20 401	C	
		compressed-data-unit	20 402	C	
		registered-data-unit	20 403	C	
205	BuiltinEncodedDataUnit	sequence-of-boolean	20 500	E	
		sequence-of-others	20 501	C	SEQ OF
206	BuiltinValue	boolean	20 600	E	
		non-negative-integer	20 601	E	
		signed-integer	20 602	E	
		real	20 603	E	
		complex	20 604	C	
		enumerated	20 605	E	
207	ComplexValue	real	20 603	E	
		imaginary	20 701	E	
208	ExternallyDefinedDataUnit	--	--	C	
209	CompressedDataUnit	data-representation	20 900	E	
		data	20 901	E	
210	RegisteredDataUnit	registration-id	21 000	E	
		pixel-values	20 101	A	
211	ExternalReference	object-address	21 100	C	
		object-format	21 101	C	
		object-internal-id	21 102	C	
		entity-address	21 103	C	
212	ExternalAddress	structured-address	21 200	C	
		cleartext-address	21 201	C	

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.		Type
301	ImageRelatedData	identifier	30 100	C	
		usage	30 101	A	
		data-type	30 102	C	CHOICE
		histogram	30 103	C	
		look-up-table	30 104	C	
		region-of-interest	30 105	C	
		neighbourhood-array	30 106	C	
		static-array	30 107	C	
		feature-list	30 108	C	
		value-bounds-collection	30 109	C	
		matrix	30 110	C	
		pixel-record	30 111	C	
		tuple	30 112	C	
		data-specified	30 113	C	CHOICE
		data-required	30 114	E	
302	Histogram	number-of-classes	30 200	E	
		class-description	30 201	C	
		classes-and-counts	30 202	C	SEQ OF
		data-specified	30 113	C	CHOICE
		data-required	30 114	E	
303	PartitionClass	lower-boundary	10 601	C	SEQ OF
		upper-boundary	10 602	C	SEQ OF
		count	30 302	C	
304	LookUpTable	number-of-input-dimensions	30 400	E	
		input-data-types	30 401	C	
		number-of-output-dimensions	30 402	E	
		output-data-types	30 403	C	
		number-of-entries	30 404	E	
		interpolation-method	30 405	C	
		input-values	30 406	C	SEQ OF
		output-values	30 407	C	SEQ OF
		data-specified	30 113	C	CHOICE
		data-required	30 114	E	
305	RegionOfInterest	roi-type	30 500	C	CHOICE
		array	30 501	C	
		ellipse	30 502	C	
		rectangular	30 503	C	
		polygon	30 504	C	
		set-of-coordinates	30 505	C	
		polarity-reversed	30 506	E	
		index-manipulation	30 507	C	SEQ OF
		data-specified	30 113	C	CHOICE
		data-required	30 114	E	

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.	Type	
306	BooleanArray	dimensionality	10 400	C	
		serialization	10 401	C	
		boolean-values	30 602	E	
307	Ellipse	roi-dimensionality	30 700	E	
		roi-size	30 701	C	
		ellipse-center	30 702	C	
		ellipse-axis-length	30 703	C	
		ellipse-dimensionality	30 704	E	
		elliptical-dimensions	30 705	C	SEQ OF
308	IntervalND	number-of-dimensions	10 500	E	
		intervals	30 801	C	SEQ OF
309	Interval1D	lower-boundary	10 601	E	
		upper-boundary	10 602	E	
310	CoordinateND	number-of-dimensions	10 500	E	
		vector-type	31 001	C	CHOICE
		integer-vector	31 002	E	SEQ OF
		real-vector	31 003	E	SEQ OF
311	SetOfCoordinates	number-of-coordinates	31 100	E	
		pixel-coordinates	31 101	C	SEQ OF
312	NeighbourhoodArray	dimensionality	10 400	C	
		serialization	10 401	C	
		element-structure	10 403	C	
		array-elements	31 203	C	
		semantic-label	31 204	E	
		key-pixel	31 205	C	
		scale-factor	31 206	E	
313	IndexND	number-of-dimensions	10 500	E	
		index-vector	31 301	E	SEQ OF
314	StaticArray	dimensionality	10 400	C	
		serialization	10 401	C	
		element-structure	10 403	C	
		array-elements	31 203	C	
		semantic-label	31 204	E	
315	FeatureList	number-of-coordinates	31 100	E	
		coordinates-and-features	31 501	C	SEQ OF
316	CoordinateAndFeature	coordinate	31 600	C	
		feature	31 601	A	

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.	Type	
317	ValueBoundsCollection	pixel-data-type	31 700	C	
		lower-boundary	10 601	C	SEQ OF
		upper-boundary	10 602	C	SEQ OF
		number-of-coordinates	31 100	E	
		coordinate-list	31 704	C	SEQ OF
318	TransformationMatrix	matrix-size	31 800	E	
		matrix-elements	31 801	C	CHOICE
		integers	31 802	E	SEQ OF
		reals	31 803	E	SEQ OF
319	PixelRecord	number-of-components	11 000	E	
		record-components	11 002	C	SEQ OF
320	PixelRecordComponent	identifier	30 100	C	
		component-value	32 001	C	
321	Tuple	number-of-elements	11 200	E	
		element-structure	10 403	C	
		values	32 102	C	SEQ OF

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.		Type
401	ImageAttribute	metric	40 100	C	
		channel	40 101	C	
		colour	40 102	C	
		control	40 103	C	
		freeform	40 104	A	
		data-required	30 114	E	
402	MetricDescription	coordinate-system	40 200	E	
		dimension-mappings	40 201	C	SEQ OF
		metric-transform	40 202	C	
		domain	40 203	C	
		data-specified	30 113	C	CHOICE
		data-required	30 114	E	
403	DimensionMapping	dimension-identifier	10 600	C	
		coordinate-axis	40 301	E	
		physical-measurement	40 302	C	
		origin	40 303	E	
		delta-type	40 304	C	CHOICE
		single-delta	40 305	E	
		multiple-deltas	40 306	C	
		precision	40 307	E	
404	MeasurementUnit	measurement	40 400	E	
		basis	40 401	E	
		verbal-description	40 402	C	
405	DeltaVector	--	--	E	SEQ OF
406	MetricTransformation	standard-transformation	40 600	E	
		matrix	30 110	C	
407	Domain	domain-type	40 700	E	
		verbal-description	40 402	C	
408	ChannelCharacteristics	band-identifier	11 600	C	
		verbal-description	40 402	C	
		precision	40 307	E	
		channel-type	40 803	E	
		channel-usage	40 804	E	
		background-value	40 805	C	
		compandor-description	40 806	C	
		sampling-function	40 807	A	
		application-specifics	40 808	A	
		data-specified	30 113	C	CHOICE
		data-required	30 114	E	

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.	Type	
409	CompandorDescription	function-type	40 900	E	
		transfer-factor	40 901	E	
		function-parameter	40 902	E	
		expander-function	40 903	C	
		application-specific	40 904	A	
410	ColourRepresentation	colour-space-type	41 000	C	CHOICE
		standardized-space	41 001	C	
		non-standardized-space	41 002	C	
		test-colour	41 003	C	
		data-specified	30 113	C	CHOICE
		data-required	30 114	E	
411	StandardizedSpace	cie-xyz	41 100	C	
		cie-yxy	41 101	C	
		cie-uvw	41 102	C	
		cie-yuv	41 103	C	
		cie-lab	41 104	C	
		cie-luv	41 105	C	
		linear-rgb	41 106	C	
		gamma-rgb	41 107	C	
		yiq-colour-space	41 108	C	
		yuv-colour-space	41 109	C	
		ycbcr-colour-space	41 110	C	
412	CIEXYZSpace	x-band-identifier	41 200	C	
		y-band-identifier	41 201	C	
		z-band-identifier	41 202	C	
413	CIEYxySpace	yl-band-identifier	41 300	C	
		xc-band-identifier	41 301	C	
		yc-band-identifier	41 302	C	
414	CIEUVWSpace	u-band-identifier	41 400	C	
		v-band-identifier	41 401	C	
		w-band-identifier	41 402	C	
415	CIEYuvSpace	y-band-identifier	41 201	C	
		u-band-identifier	41 400	C	
		v-band-identifier	41 401	C	
416	CIELabSpace	l-band-identifier	41 600	C	
		a-band-identifier	41 601	C	
		b-band-identifier	41 602	C	
		white-point	41 603	C	

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.	Type
417	CIELuvSpace	l-band-identifier	41 600	C
		u-band-identifier	41 400	C
		v-band-identifier	41 401	C
		white-point	41 603	C
418	CIEXYZCoordinate	cie-x	41 800	E
		cie-y	41 801	E
		cie-z	41 802	E
419	LinearRGBSpace	representation	41 900	E
		illuminant	41 901	E
		r-band-identifier	41 902	C
		g-band-identifier	41 903	C
		b-band-identifier	41 602	C
420	GammaRGBSpace	representation	41 900	E
		illuminant	41 901	E
		r-band-identifier	41 902	C
		g-band-identifier	41 903	C
		b-band-identifier	41 602	C
421	YIQColourSpace	y-band-identifier	41 201	C
		i-band-identifier	42 101	C
		q-band-identifier	42 102	C
422	YUVColourSpace	y-band-identifier	41 201	C
		u-band-identifier	41 400	C
		v-band-identifier	41 401	C
423	YCbCrColourSpace	y-band-identifier	41 201	C
		cb-band-identifier	42 301	C
		cr-band-identifier	42 302	C
424	NonStandardizedSpace	rgb	42 400	C
		ihs	42 401	C
		cmy	42 402	C
		cmyk	42 403	C
		n-band	42 404	C
425	NonStandardizedRGB	r-band-identifier	41 902	C
		g-band-identifier	41 903	C
		b-band-identifier	41 602	C
		white-point	41 603	C
		primaries	42 504	C

Table 7 - List of IIF-DF syntax entities and components (cont'd)

IIF syntax entity		Component names	No.	Type	
426	NonStandardizedIHS	i-band-identifier	42 101	C	
		h-band-identifier	42 601	C	
		s-band-identifier	42 602	C	
		white-point	41 603	C	
		primaries	42 504	C	
427	Primaries	r-primary	42 700	C	
		g-primary	42 701	C	
		b-primary	42 702	C	
428	CIExyCoordinate	cie-x	41 800	E	
		cie-y	41 801	E	
429	NonStandardizedCMY	c-band-identifier	42 900	C	
		m-band-identifier	42 901	C	
		y-band-identifier	41 201	C	
		verbal-description	40 402	C	
430	NonStandardizedCMYK	c-band-identifier	42 900	C	
		m-band-identifier	42 901	C	
		y-band-identifier	41 201	C	
		k-band-identifier	43 003	C	
		verbal-description	40 402	C	
431	NonStandardizedNBand	number-of-bands	11 500	E	
		band-descriptions	43 101	C	SEQ OF
432	ColourBand	band-identifier	11 600	E	
		verbal-description	40 402	E	
433	TestColour	colour	40 102	C	
		region	43 301	C	
434	PIKSControl	roi	43 400	C	
		roi-offset	43 401	C	
		match-point	43 402	C	

Table 7 - List of IIF-DF syntax entities and components (cont'd)