DIG35

– Metadata for Digital Images –

Working Draft, 1.0

March 6, 2000

Digital Imaging Group

About the Digital Imaging Group

The Digital Imaging Group (DIG) is a not-for-profit imaging industry consortium of manufacturers, developers, and users of digital imaging solutions. Originally founded in 1997 by Adobe, Canon, Eastman Kodak, Fuji, Hewlett-Packard, IBM, Intel, Live Picture and Microsoft, the organization has grown to over 70 members at three different levels of membership as of Jan. 17, 2000. The primary goal of the Digital Imaging Group is to provide an open industry forum for the exploration, implementation and stewardship of technologies and methods for expanding the Digital Imaging market.

For more information:

Digital Imaging Group, Inc

http://www.digitalimaging.org

Comments should be sent to:

dig35comments@digitalimaging.org

Members of the DIG35 Initiative Group are (in alphabetical order):

Adobe Systems Inc., AGFA-GEVAERT N.V., Canon Inc., Digitella Technology Inc., Eastman Kodak Company, Fuji Photo Film Co., Ltd., Hewlett Packard Company, Microsoft Corporation, NETIMAGE, PhotoChannel Networks Inc., PictureIQ Corporation, Polaroid Corporation, Seattle FilmWorks, Inc.

Foreword

From consumer to professional and commercial applications, digital images are overtaking the traditional images created with film or video in terms of number of images created as well as becoming closer in quality of image. In addition the increasing sales of digital image capture devices and associated services available for digital image processing will soon result in the proliferation of huge numbers of digital images in many forms and in a variety of sizes and [perceived and measured] value.

From the very beginning of photography (and before that, in art of all kinds) those who take pictures have tried to capture as much information about the photograph as possible. This information is termed *metadata*.[i]

Every family has shoeboxes filled with photographs, some with a few cryptic notes on the back. Another method of tracking metadata has been in photo albums dedicated to special occasions. In both cases the information has been limited and often the meaning of the notes is lost over time. Creating metadata for images enhances our memory and increases the joy of sharing our memories. As we share images across distance and time it becomes even more important to remember as much as possible about the image.

In today's digital age, sharing images is easier than ever. The growth in digital technology has created the desire among users to manage and exchange their images in a variety of ways including storage, e-mail exchange, Internet/WWW postings and other displays (such as personal electronic photo albums or photo frames), and printing to a variety of devices with different resolutions.

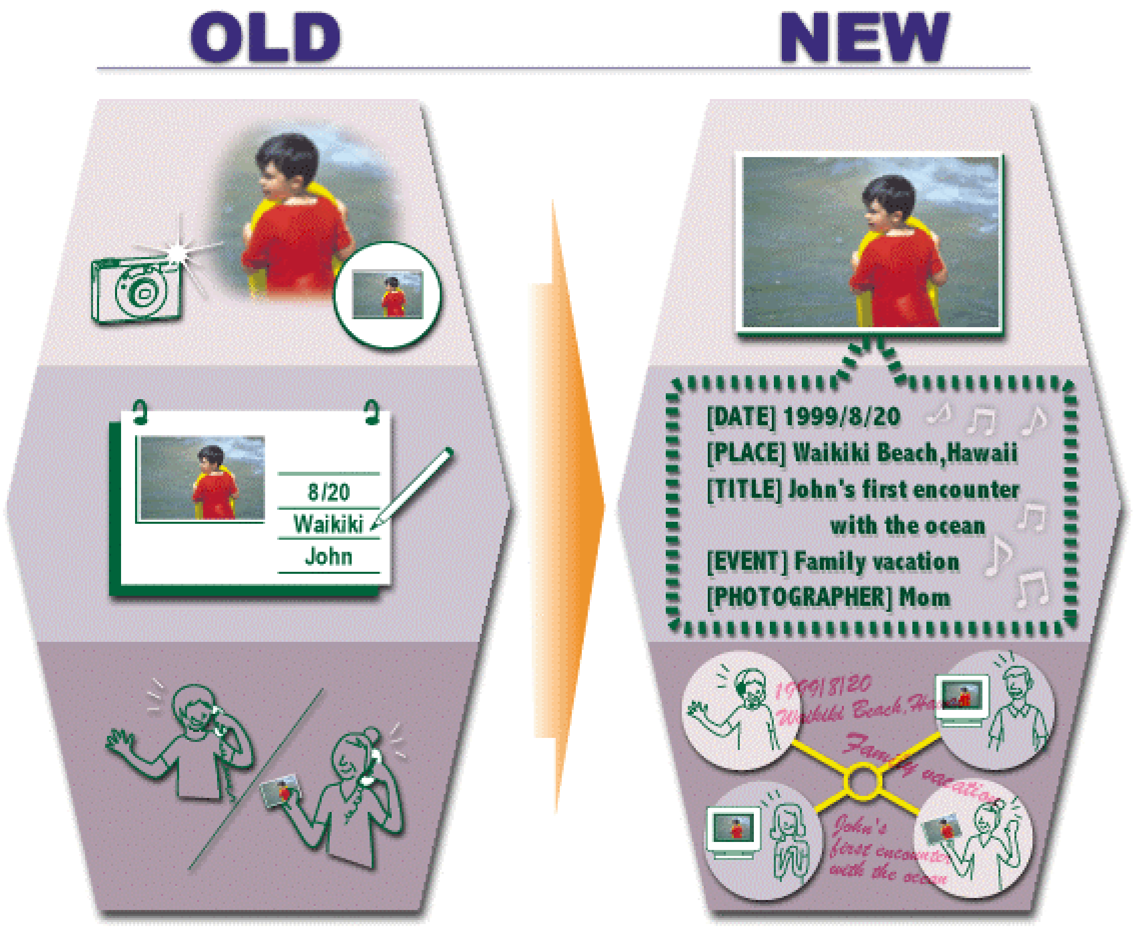
The illustration below highlights several of the key enhancements in using today's technology and the benefits of having more extensive metadata.

First, with additional metadata the image becomes important not just for today but also in the future. Knowing exactly who, what, when, where, and how a photograph was taken provides a solid basis for documenting our lives or our business.

Second, using today's technology, images can be shared far more broadly and in the way most appropriate to each individual. This extensive sharing of images requires more knowledge about the image, especially in a business situation.

And, finally, knowing as much as possible about the image enhances the joy of sharing. We no longer need to rely on memory alone to provide the background for our memories.

[i] The term metadata is composed from the Greek word "meta" meaning aside and "data" meaning the essential content of an electronic container (for example, an image file).



Digital technology, whether by digital cameras or by digitizing video or film-based images, paves the way for many more of these associated services than would be available solely within analog media such as traditional film or VHS video. One of the primary services becoming more broadly available in the computerization of images is that of enhanced metadata. Today's image and data processing systems provides the means to have an image (the primary data under consideration) be associated with much more of the important information that is not part of the primary but nonetheless very valuable.

Contents

List of Figures

List of Tables

1 Introduction

Metadata, to be most valuable for the owner(s) and user(s) of an image, needs to be consistently maintained throughout the image lifecycle. In today's environment of image editing applications, rapid transmission via the Internet, and high quality photographic printers, the lifecycle of a digital image may be very long as well as complex.

However, if standards for managing and exchanging these images, including associated data input methods for users, do not materialize, we will be left with the digital equivalent of the "shoebox" and will not realize the full potential of digital imaging power and flexibility.

Realizing this the Digital Imaging Group formed an Initiative Group in early 1999 to begin addressing the issues involved and provide the first step towards a standard set of metadata for digital images. This Initiative Group (internally called DIG35) includes representatives from Adobe Systems Inc., AGFA-GEVAERT N.V., Canon Inc., Digitella Technology Inc., Eastman Kodak Company, Fuji Photo Film Co., Ltd., Hewlett Packard Company, Microsoft Corporation, NETIMAGE, PhotoChannel Networks Inc., PictureIQ Corporation, Polaroid Corporation, Seattle FilmWorks, Inc.

The result of this broad collaboration of imaging and technology industry participants is this technical specification defining specific metadata for digital images and a recommended implementation model. In total this set of documents comprises the "DIG35 Specification for Digital Image Metadata" or DIG35 specification.

1.1 Vision and Goal

The vision of the DIG35 is:

> “To provide a mechanism to allow end-users to see digital image use as being equally as easy, as convenient, and as flexible as the traditional photographic methods while allowing benefits that are possible only with a digital format.”

The overall goal of the DIG35 initiative is to define a standard set of metadata for digital images that will improve the semantic interoperability between devices, services and software, thus making it easier to organize, print, exchange and otherwise process digital images in this new millenium.

1.2 Scope

This specification defines a set of ***public metadata*** for digital still images, either as a single image or a collection of images, that could be supported and exchanged by current and future devices, software, and services in an open environment. In addition to the metadata definition, a recommendation for implementation is defined to enable exchange of such metadata.

This specification does not concern itself with vendor specific/proprietary *private metadata* or a particular application domain. Private or application domain specific metadata should be defined within their respective organizations and is out of scope of this specification.

This specification also does not mandate procedural metadata[i] such as processing parameters that changes the visual appearance *after* decoding the image data (e.g. cropping, rotating etc.), printing information (e.g. number of copies, output size etc.) and ordering information (e.g. delivery and payment information). There are, however, exceptions at this time. Print Aspect Ratio (PAR) metadata is an example that does not follow this rule. Technical metadata that are well-defined and intended to produce better quality images that benefit the users without additional user effort are considered exceptions.

[i] Such metadata are either discussed in other organization or subject to future discussions and efforts by the Digital Imaging Group.

1.3 Design Requirements

The design goals and principles of this specification are identified below to give a high-level direction of this specification.

1.3.1 Design Goals

The design goals of the DIG35 initiative are to define a metadata set that is:

- INTERCHANGEABLE: DIG35 is based on a sound conceptual model that is both generally applicable to many applications and assured to be consistent over time. DIG35 will create a better environment for locating and (re)using specific images.
- EXTENSIBLE AND SCALEABLE: DIG35 enables application developers and hardware manufacturers to add additional metadata fields. This allows future needs for metadata to be fulfilled with limited disruption of current solutions.
- IMAGE FILE FORMAT INDEPENDENT: DIG35 does not rely on any specific file format and can therefore be supported by many current and future file formats and compression mechanisms.
- CONSISTENT: DIG35 can work well with existing standards today allowing the metadata standard to be widely acceptable and usable in a variety of application domains and user situations.
- INTERNET-READY: DIG35 provides seamless integration with the Internet by utilizing XML (the recommended implementation method).

1.3.2 Design Principles

Design principles adhered to when developing the DIG35 initiative were developed prior to beginning the technical efforts. These principles state that the DIG35 initiative shall:

- use existing standards and output of other organizations as much as possible while creating a future-looking metadata standard.
- focus on mid and long-term perspectives, not only on what current digital imaging technology may be able to offer today
- be simple for developers to utilize but sophisticated enough to cover a wide spectrum of features
- support *information* preservation, not *data* preservation
  Many devices, such is digital cameras, may store metadata in a format that end users are not familiar with which will discourage use. Thus, applications may need to apply appropriate conversions to transform these values into user-understandable formats.
- allow metadata redundancy
  While values exist that can be calculated from other fields, at the definition level, redundant metadata do exist and need to be managed appropriately.

1.4 Organization of This Document

[Editor's note: To be completed when the draft is fixed.]

1.5 Conventions

Syntax, and code examples are shown in `fixed width` font.

Italic strings are used for emphasis and to identify words representing variable entities in the text.

Bold strings are used to identify the first instance of a word requiring definition; see the Glossary for definitions.

Links are shown in colored fonts

NOTE: Text that is of special interest is represented as a note to the reader.

1.6 Status of This Document

This document is a [working draft] recommendation of the DIG35 Initiative Group. Review comments on this document should be sent by March 31, 2000 to <`dig35commnets@digitalimaging.org`>.

This is a working draft version document and may be updated, replaced or obsoleted by other documents at any time.

2 Metadata Architecture

2.1 What is Metadata?

As previously discussed, metadata is additional information that is associated with the primary data (the image). In the context of this specification, it is "additional data linked with the image data beyond the pixels which define the image." Metadata may be used in a variety of ways, including:

- Providing in-depth information on the image and its creation, such as date, time of day, focus distance, light levels, use of flash, GPS location, etc.
- Allowing easy indexing, identification, categorization and usage-control according to any pre-determined schema, such as image type, copyright conditions, originator, subject matter, location, etc.
- Enhancing the intrinsic content of the image such as differentiating between several different beach images, which may look the same to an unknowing observer.

Example metadata:

Photographer: Kats Ishii

Date: 1999/12/09

Location: Maui, Hawaii

Title: View from Conference Room

Comment: A nice day to swim!

Event: ISO meeting

Figure 1: Sample image with metadata

2.2 Image Model

Digital images are commonly stored in well defined file formats, e.g. JFIF(JPEG), Exif, TIFF, Flashpix, or others. Although there are differences in how the data is physically stored on the recording media, conceptually each format has a similar Image Model. The illustration below shows the logical structure of several of today's digital image formats. Note that each file format, while taking a unique approach, consists of three major components: the header information, the image data, and metadata.

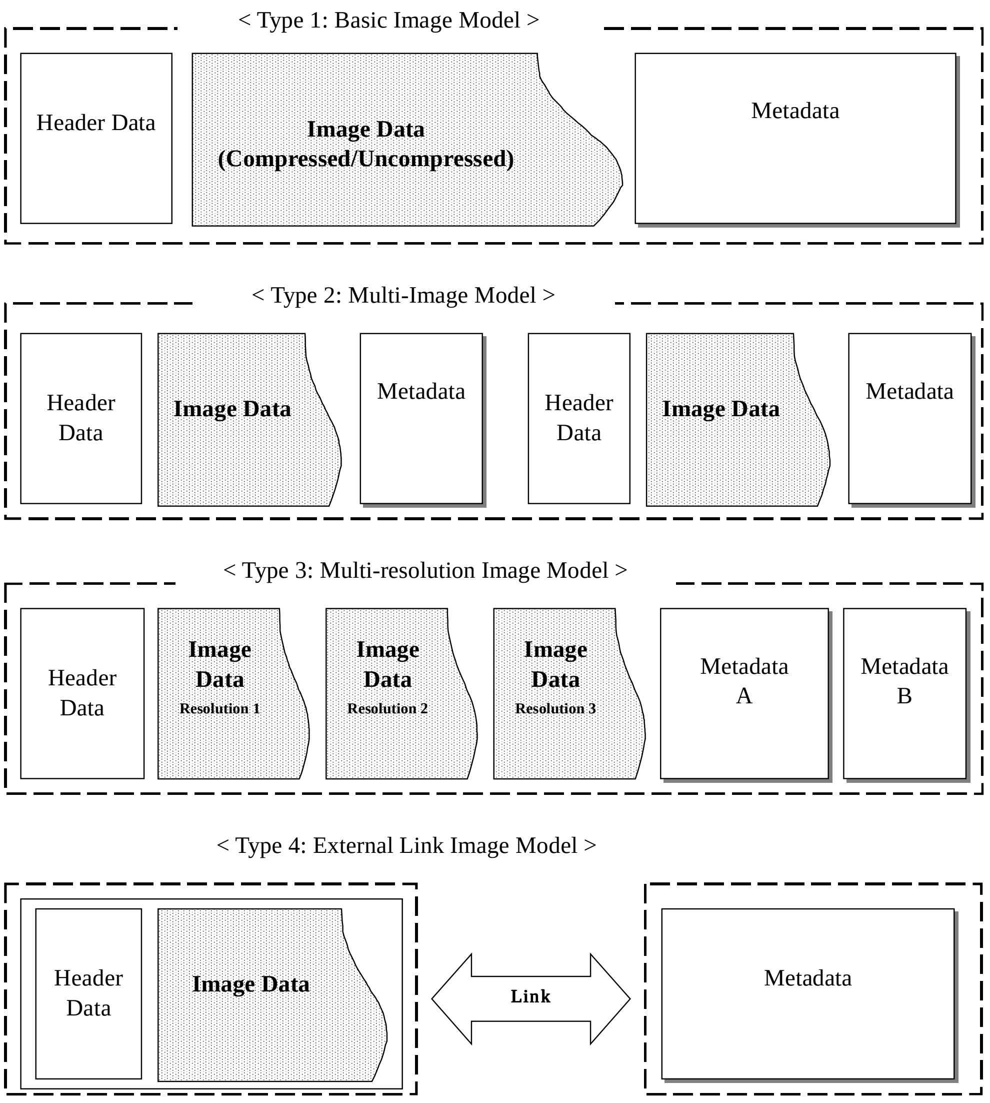


Figure 2: Several examples of the conceptual structure of a Digital Image

[Editor's note: Type 1 is the reference model used throughout the rest of this document.]

The "Header Data" is a specific set of metadata that consists of essential information to decode the image data (such as the image size, bit depth, colorspace and compression method, etc.) plus additional information required to locate the image data and the other metadata within the file. While this information is metadata of the image data by definition, it is tightly coupled with the image data and specific to a particular image file format. Thus the detail definition of the "Header Data" is considered a mandatory entity of the image file format. As previously noted, the definition of file format is not the scope of this specification.

The "Image Data" is the primary data of a digital image[i]. It is the digitally encoded representation of a picture that is either compressed or uncompressed, raw signal data of the capture device. Other related data, either directly or indirectly, are considered as "metadata" in this specification.

Other non-image data are termed as "Metadata" and may include information such as the capture conditions of the camera, a description of the image content, and intellectual property information. The built-in capability of storing metadata within an image file format varies from format to format. Variants of a bitmap format, for example, do not define placeholders for metadata.

[i] Other file format specifications may term it as either codestream or bitstream.

JFIF (JPEG), on the other hand, defines a mechanism to allow metadata to be associated with the image data. However JFIF (JPEG) does not currently define the type of metadata to be stored. This means that the storage mechanism and metadata structure must be defined by each application in order to exchange metadata associated with that file. Lastly, Exif, TIFF, SPIFF and Flashpix are examples of file formats that define a rich set of metadata along with a storage mechanism. Defined metadata in these file formats includes descriptional information such as the title, date/time of the capture, the capture source (e.g. camera, scanner) from which the image was created, and copyright information. There is compatibility of these file format based metadata definitions to some extent; however, not all metadata can be mapped from one format to another. This lack of direct mapping causes a variety of problems when conducting a simple image file format conversion.

In addition to the metadata that is closely associated with the file format image data, there are other metadata that are used throughout the lifecycle of the image data. Such metadata may include print condition information such as output media size, the number of copies and service order information (i.e., delivery instructions), and payment methods. While this information may be very useful in various applications, it is out of scope of this specification.

2.3 Metadata Interchange Model

2.3.1 Image Pixel Data Interchange

The most important aspect of digital imaging is to be able to view, share, and enjoy the actual image in a variety of ways. Those images are, in most cases, captured directly with a digital camera or created by scanning a film or prints. These digital images are stored as an image file on a storage media to be used in various ways to fulfill the objectives of the user. Typical uses are, for example, opening an image into an application to be printed for personal pleasure or used to create post cards and/or web pages to share them with family members or friends. The fundamental value to the user is the capability of viewing the captured picture in a variety of ways and at different times.

This type of image data interchange and work flow are enabled due to the fact that current devices and applications support an image file format model which is well defined and standardized. Thus the actual image data can be interchanged between various file formats.

Figure 3 shows a typical model of image data interchange. Filters include image libraries or plug-in type applications that enables the system to read and store the image data into it’s native (proprietary) data format. As long as both systems support the file format model, the image data can be successfully interchanged.

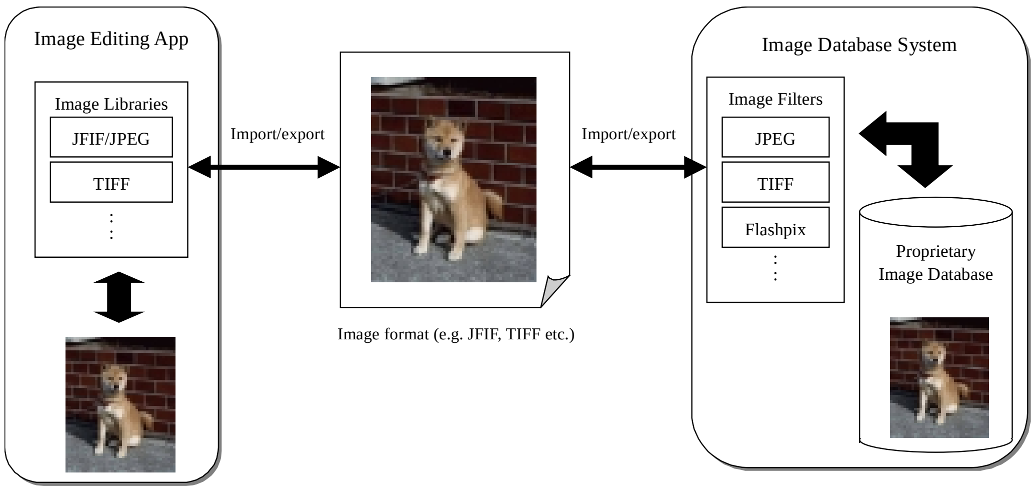


Figure 3: Image data interchange model

2.3.2 Image Pixel and Metadata Interchange

While it is true that devices and applications support image data interchange, the metadata that is to be associated with those images are not always passed on within the imaging work flow. In application terms, the persistence of metadata is not implemented. There are many reasons for this lack of support by applications. Among the most important ones are:

- The file format does not support metadata that the application uses.
- The metadata is stored in a proprietary file format that is not publicly defined.
- A standard image metadata definition and interchange format model does not exist.

Therefore, users are not able to reuse the metadata that have been automatically recorded by the capture devices or the information that they may have entered. Once a file format independent metadata definition and interchange format is standardized, users will be assured that those metadata previously entered are properly retained and treated as the important information that it represents. Users will thereby be encouraged to input more metadata and gain more benefit from the superior technology digital imaging offers.

Thus this specification consists of two major parts: Part 1, the metadata definition and Part 2, the reference implementation for interchange. The specification in its entirety is designed based on the following principles, which encompass vendor-neutrality and technology-independence.

- Partition the metadata definition into logical sub-blocks; devices with limited resources may choose to support portions of this definition and those applications that need further detail can easily extend it's scope
- Use a widely accepted, flexible and extensible implementation format (XML)

The metadata interchange model is shown in

Figure 4. Similar to that of the image data interchange model, metadata is also imported/exported via a standard metadata reader/writer ability (in the case of the reference implementation of this specification, an XML parser) that can map the information into its system's database/data structure. It is ideal for imaging tools to share a common metadata model within each application or system that will enable easy sharing of metadata from one application to another. However, given the diversity of digital image devices today and in the foreseeable future as well as applications in a wide range of spectrum, it is neither feasible nor realistic to standardize a single unified internal metadata model that is sufficient for all purposes.

Note: there are additional challenges attributed to associating metadata with images and the problems that are encountered relating to retention of image metadata throughout the lifecycle of that image. (See Appendix II for detail discussions.)

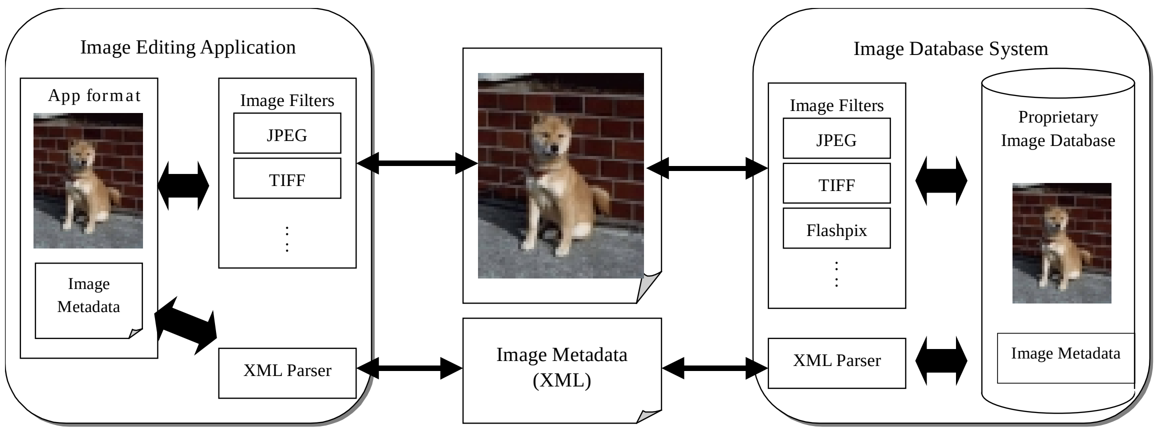


Figure 4: Metadata interchange model

2.3.3 Challenges of Metadata Interchange

Vendors and users alike recognize the value of image metadata; however, the rapid proliferation of data manipulation and management tools has resulted in a wide variety of information technology products that process metadata differently, or not at all and without much consideration for sharing or exchange of metadata. Thus the challenge is to define a standard set of metadata that is generally acceptable by vendors, users and content providers. The challenges include:

- creating a well defined semantic model that promotes interoperability of metadata
- creating proper rules and guidelines for retention and association of metadata with image data
- educating users and providing guidelines

2.4 Metadata Storage Model

A special case of image interchange is image storage. For image storage, the metadata could be (and perhaps typically is) separated from the image data. The interchange file would be "burst" and the metadata might populate a relational database. This raises several interesting challenges for the management of images, but also offers significant advantages.

Once burst, the image management system can optimize how it manages the image and metadata individually. The most obvious example of this is the difference in physical size of the data. Image data might be stored on a staged archival access mechanism (disk → tape), while the metadata might be stored on magnetic. Another example would be to optimize around how the data is accessed and used. If the metadata is only used after an image is accessed, it could reside on staged access media along with the image. If the metadata is used for image search, it could be loaded into a Relational or an Object Oriented database. In some cases, a mixture of these approaches could be used.

However, once burst, the image management systems has lost the physical relationship between the image and metadata. If it becomes necessary to access an exact replica of what was received, this may be difficult. Whether this is important or not is yet to be determined in the marketplace. However, there is significant effort being made to protect images using watermarks and encryption technology. File formats are beginning to make use of IPR (Intellectual Property Rights) tags to indicate who owns or created images. If these properties apply to the collective image and metadata it becomes a requirement to track these items as a group. This could be difficult once a file is burst.

2.5 Progressive Metadata Model

Most closely aligned with the capabilities of progressive compression and file formats such as JPEG 2000, the progressive metadata model allows a client to progressively access metadata as the client needs the additional information. This a is an extension of storage metadata case that allows the user to only access the metadata that is needed for a particular purpose. As in the "burst" storage case, the integrity of the original image has been lost, and a particular user may never have the complete image metadata set and may not be aware of this fact.

3 DIG35 Image Metadata Definition

3.1 Overview

Image metadata is a “building block” for digital imaging that may be used within the wide spectrum of the imaging work flow. This specification defines a standard set of image metadata based on a generic concept that may be further divided into conceptual “sub-blocks.” Each of these sub-blocks describes a unique aspect of the image. By partitioning metadata into discrete sub-blocks, developers may extend a particular sub-block without affecting the entire architecture thereby ensuring semantic interoperability while allowing vendors to add value to the metadata and image data itself. The DIG35 metadata sub-blocks are shown in Figure 5.

Note that procedural metadata are out of scope of this specification. Examples of procedural metadata include;

- Parameters for image processing that change the visual appearance by arbitrary cropping, rotation, or other transformations.
- Order information to specify product and quantity.
- E-commerce information such as billing or payment data or delivery addresses.

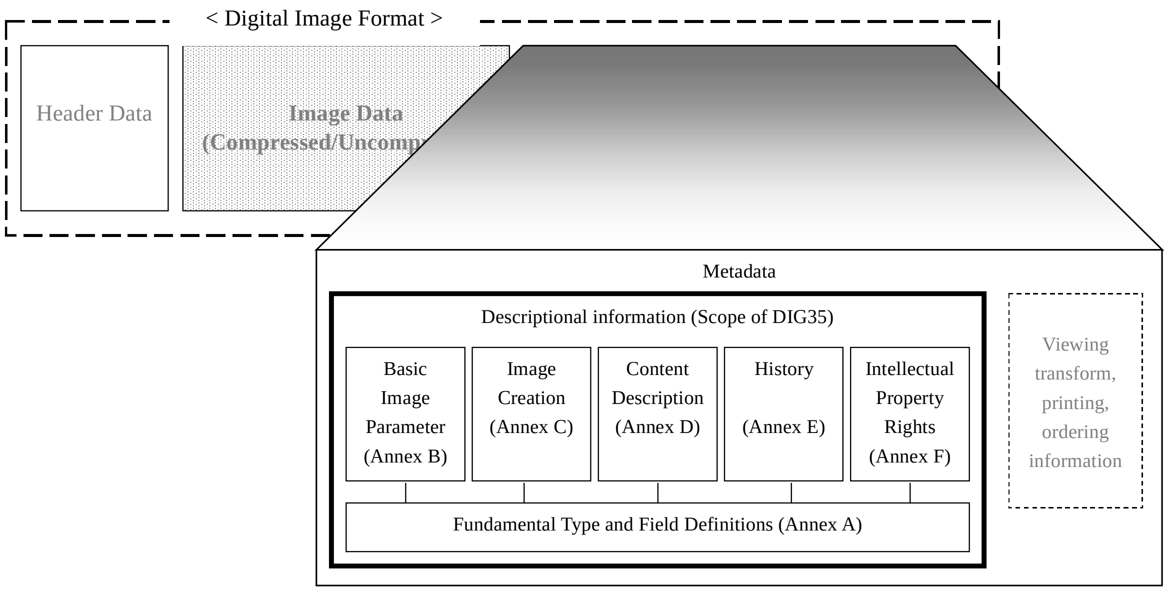

< Digital Image Format >
Header Data
Image Data
(Compressed/Uncomp
Metadata
Descriptional information (Scope of DIG35)
Basic Image Parameter (Annex B)
Image Creation (Annex C)
Content Description (Annex D)
History (Annex E)
Intellectual Property Rights (Annex F)
Fundamental Type and Field Definitions (Annex A)
Viewing transform, printing, ordering information

Figure 5: Scope of this specification within the Image Model

3.2 DIG35 Metadata Blocks

DIG35's Metadata definition consists of five logical sub-blocks with a separate common definition that is referred to by other sub-blocks. While each sub-block is logically partitioned, they may be linked to each other to form additional semantics.

3.2.1 Fundamental Metadata Types and Fields

The Fundamental Metadata Types define the format field defined in all metadata sub-blocks. Those may include a collection of metadata elements such as an address or a representation of an attribute of other elements such as the language.

The Fundamental Metadata Fields define fields used in all metadata sub-blocks. These include a definition for language specification and timestamps.

3.2.2 Basic Image Parameter Metadata

Since this specification is a general-purpose metadata standard, it must be applicable to the broadest possible class of file formats. Since each file format makes distinct decisions regarding what elements are important as header information, it is impossible to delegate the specification of header metadata to file format developers. In fact, this specification takes the opposite approach and assumes the existence of a file format that contains no header information. This assumption ensures that any format may be transcoded into another file-format.

In order to do this, a block of metadata is defined that contains information similar to, and identical in use to, file header metadata. There should never be conflicts between this sub-block and the file header metadata as this sub-block is intended to be used, as stated above, *only* when there is no file header metadata. However, if there is a conflict between the file-format header information and the Basic Image Parameter Metadata, the file header should always take precedence.

Note: Each file format defines its own header information that is very dependent on the features the format supports. Those features are specific on the image data and thus out of scope of this specification. This metadata should be considered informational and not be used to decode the image data stored in the associated file.

3.2.3 Image Creation Metadata

The Image Creation Metadata defines the "how" metadata that specifies the source of which the image was created. For example, the camera and lens information and capture condition are useful technical information for professional and serious amateur photographers as well as advanced imaging applications.

3.2.4 Content Description Metadata

The Content Description Metadata defines the descriptive information of "who", "what", "when" and "where" aspect of the image. Often this metadata takes the form of extensive words, phrases, or sentences to describe a particular event or location that the image illustrates. Typically, this metadata consists of text that the user enters, either when the images are taken or scanned or later in the process during manipulation or use of the images.

3.2.5 History Metadata

The History Metadata is used to provide *partial information* about *how* the image got to the present state. For example, history may include certain processing steps that have been applied to an image. Another example of a history would be the image creation events including digital capture, exposure of negative or reversal films, creation of prints, transmissive scans of negatives or positive film, or reflective scans of prints. All of this metadata is important for some applications. To permit flexibility in construction of the image history metadata, two alternate representations of the history are permitted. In the first, the history metadata is embedded in the image metadata. In the second, the previous versions of the image, represented as a URL/URI, are included in the history metadata as pointers to the location of the actual history. The history metadata for a composite image (i.e., created from two or more previous images) may also be represented through a hierarchical metadata structure. While this specification does not define the "how" or "how much" part of the processing aspect, it does enable logging of certain processing steps applied to an image as hints for future use.

Note: Neither the processing nor the history metadata specify the actual processing steps or the compositing steps. Such metadata is reserved for further study.

3.2.6 Intellectual Property Rights (IPR) Metadata

The Intellectual Property Rights Metadata (IPR) defines metadata to either protect the rights of the owner of the image or provide further information to request permission to use it. It is important for developers and users to understand the implications of intellectual property and copyright information on digital images to properly protect the rights of the owner of the image data.

Note: Several international organizations are looking at standards in the area of intellectual property rights.

3.3 Example Work Flow

The work flow of digital imaging varies from that of the analog image processing and work flow. Some of the differences in the two work flows are highlighted in the examples below. The DIG35 use-cases provide additional insight into the digital image work flow issues that DIG35 are concerned with in this specification.

Note: See companion document "DIG35 Requirements and Use-cases" (in preparation) for further detail of DIG35 use-cases.

3.3.1 Example 1

A typical imaging work flow including metadata retention consists of three basic processes; Creation, Management/Manipulation and Output. An example work flow is illustrated in Figure 6 that shows the type of metadata that could be added to the image data. By persisting the metadata recorded in the previous process, the next application or device would be able to take advantage of those information and use them appropriately without involving duplicate user inteactions. The dotted line denotes the metadata that are in scope of this specification.

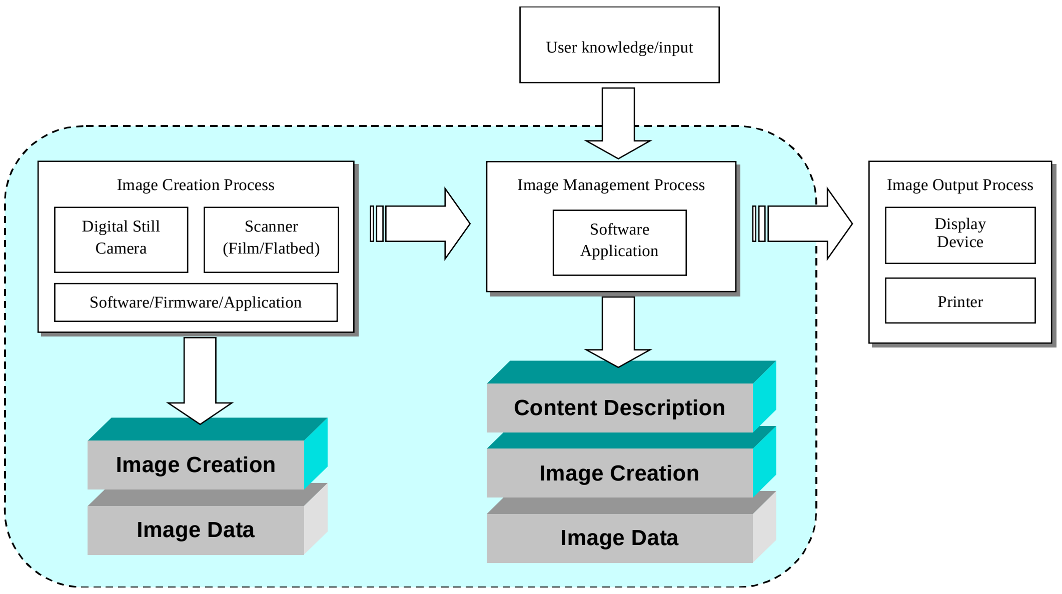


Figure 6: Example image work flow by process

3.3.2 Example 2

The following figure (Figure 7) shows interoperability of several applications that all process the same digital picture with metadata.

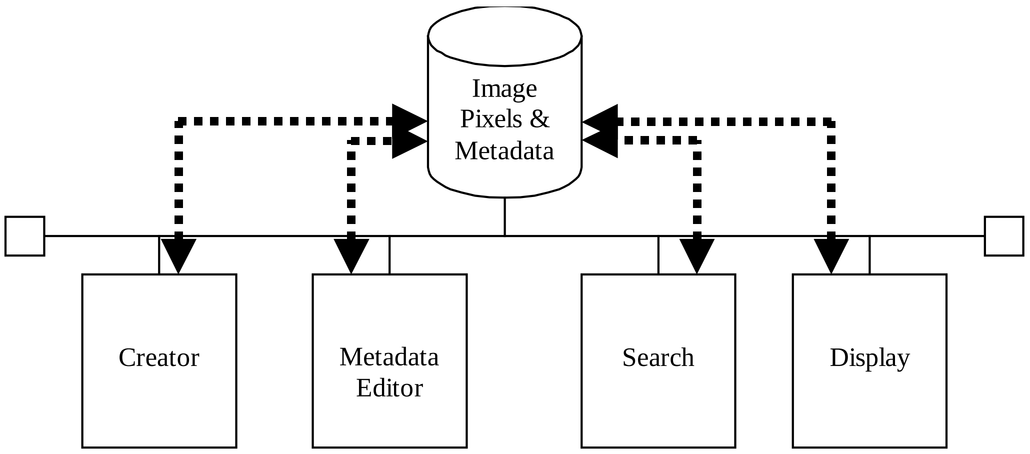


Figure 7: Different metadata access/usage example

The heavy dashed lines indicate information flows accessing the metadata. The same metadata must work with all of the applications, for example creation, editing, search, and display

The disk icon represents a single picture file, with metadata. The applications represented are typical, but certainly not limited to the particular types of applications shown. The heavy dashed lines represent the information flows that allow the applications to get and set the metadata. In this example, the picture file resides in a single computer. The applications may run on the same computer, or may run on different computers with networked drives. This means that the information flows are not protocol-based interchanges. Instead they represent application programming interfaces allowing the programs to access the metadata. It is assumed that the applications run one at a time, so there is no need to for record locking and associated complications.

The applications each read and parse the XML data. This builds a model of the metadata in the application memory. The application may just read the metadata, and not modify it. Alternatively, an application may update or delete existing metadata, and may create additional metadata. Periodically (or else just before exiting) the application updates the picture's metadata. To do so, it creates the XML from the internal metadata model stored in its memory, and then writes the XML to the file.

This promotes interoperability of the same data file among and between several independent applications, all based on the common information model and format of the DIG35 metadata.

4 Implementation Definitions

[Editor's note: The actual XML definitions will be in Annex G and H.]

4.1 Using XML

This specification recommends using the World Wide Web Consortium (W3C) XML (Extensible Markup Language) as the standard metadata interchange format. While other implementation methods are certainly possible, the primary reasons for choosing XML include:

- XML is already widely adopted as a cross-platform and Internet-enabled implementation language;
- Many applications within the imaging workflow can interface to XML structures;
- XML provides a highly extensible method for creating device-independent, language-independent, and application-independent interchange formats;
- XML is equally well-suited for handling relational or hierarchical data structures;
- XML provides a solid foundation for implementing both human- and machine- readable and understandable metadata.

Because the industry has already developed a robust set of tools for both writing and reading XML, it offers a ready-made environment which developers of digital imaging applications and devices can rapidly incorporate the advantages of metadata structures. In addition, the widespread usage of XML within the Internet community provides for an inherently smooth meshing of the metadata structures with underlying network transport mechanisms and Web-based communication methods.

4.1.1 DTD or XML Schema?

Currently the content description is being developed as an XML DTD (Document Type Definition).

XML Schema offers benefits over a DTD including:

- Type information for data (eg. Integer / Float etc)
- Type inheritance

XML Schema is not yet a standard and as thus has not been used in this specification. It is strongly recommended that once XML Schema becomes standardized, it be adopted as the method for specifying the syntax and data structure of the DIG35 metadata.

4.1.2 Multi-Language Support

Note that XML is usually stored as either UTF-8 or UTF-16, both of which can store multiple languages (regardless of the character set) in a single document. As the use of digital images provides the potential to work together seamlessly, metadata will also need to be provided that can be read and understood in many languages and character sets. For example, names of the people in the image may be in German, Russian, Japanese, English, and Chinese (for a university cooperation program).

5 Miscellaneous Topics

5.1 Association of Metadata and Image Data

There are many challenges attributed to metadata usage within the imaging work flow. After conforming to a standard metadata set, the next most significant challenge is associating those metadata with the image throughout a standard work flow. There are several alternatives — each having benefits and drawbacks that may create new problems to be encountered relating to retention of image metadata throughout the lifecycle of that image. A detailed analysis of metadata association and possible metadata association methods is discussed in Appendix II.

5.2 Redundant Metadata

5.2.1 Redundant Data Within the Metadata

Some of the metadata fields may cause redundant data to be stored. For example, if the date an image taken at a birthday party, the subject's birthdate, and the subject's age are all stored as metadata, then there is the potential that these three fields may conflict.

Calculation of missing redundant data is certainly expected. For example, if a search engine is searching for images of people approximately 40 years old, then calculating their age from the image date and the birthdate is possible. Such interaction of fields is beyond the scope of this specification.

If redundant information is inconsistent, the file should still be treated as valid. There is a possibility that calculated information produces an inconsistent (in error) value. This specification does not define the mechanism for solving this situation. The DIG or others may provide "Best Practices" documents to help developers and users with such situations.

5.2.2 Redundant Data Between Metadata and Image Data

Where metadata duplicates the image data (for example the number of pixels may be stored in the metadata and in the image itself), there is the possibility of redundant and inconsistent data. Where there is redundant data (such as in the basic image parameter sub-block), the metadata should not be used to decode the image. The metadata is useful, especially in a system where the overhead (time and/or complexity) is large and the image is not interpreted, values stored in the metadata can still be used to classify the image. For example: A user may specify to only search for high-resolution images.

NOTE: The decision on whether to recognize the metadata or the image data as the correct data when there is redundancy is still under investigation. DIG35 welcomes comments in this area.

5.3 Inconsistent Metadata

Each of the metadata fields can contain a timestamp. This timestamp can be used to determine whether the metadata is newer or older than the image data. Where the metadata is newer than the image data, then the metadata can be assumed to be consistent.

Where the image data is newer than the metadata, then the metadata may not be consistent with the image, and especially fields like the location need to be checked for accuracy before use.

5.4 Metadata Compression

Significant time and effort have been invested by many standards organizations and commercial developers to create efficient compression algorithms for digital images. However, compression of metadata has not been explored and developed to the same depth.

In general, high-quality image data is magnitudes greater in size than the metadata associated with it. Yet when the image size is reduced, for example to represent a thumbnail, the size ratio is reversed. Size of the metadata quickly becomes an issue for both users and developers when this occurs.

DIG35 has identified requirements for metadata compression systems. One of the recognized requirements from the DIG35 use-cases is that file formats that are able to store compressed metadata must store header information that describes the compression algorithm and any necessary decompression parameters. This is to ensure that a program reading the metadata as a byte stream has the minimal information required to make use of the metadata *before* it actually sees the metadata. An additional requirement recognized is that file formats that store compressed metadata must compress significant blocks of metadata *en bloc* and note create a large number of small, independently compressed segments.

Several ad-hoc experiments have demonstrated significant compression in moderately-size XML metadata is possible using currently available compression algorithms. Reduction in metadata sizes of 50 – 70 % has been observed using one widely-available commercial compression package. The Wireless Access Forum (www.wapforum.com) has defined a wireless access protocol (WAP), which includes a component for binary representation of XML. This technique may also yield significant metadata data compression.

Based on these experiments and observations, it is premature to select a particular compression scheme. However, these results are conclusive enough to determine that a recommended practice is necessary rather than any additional compression algorithms defined specifically for metadata. Such definition is beyond the scope of this specification.

NOTE: DIG35 is investigating the potential inclusion of a "null compression" algorithm which would be defined so that all metadata would have the compression algorithm, version, and parameter identification metadata for consistency. DIG35 welcomes comments in this area.

5.5 Security Considerations

The full spectrum of possible security risks associated with image metadata is beyond the scope of this specification. However, this section addresses the importance of authentication of metadata and of verification that the pixel data and metadata are properly associated.

There are a number of security risks associated with metadata. The security mechanisms for authentication and privacy address some of these risks. The tolerance level for risk varies widely, as does the likelihood of a security violation. The typical consumer digital images are not likely to need security methods as extensive as those of a governmental or commercial image bank. In many consumer instances, the concern is not of overt (or covert) threats to their images and metadata, but rather a concern that there might be accidental changes to metadata, or that an images is incorrectly associated with metadata from a different image.

Some applicable security mechanisms include:

- Hash generation—a technique that creates a hash value from given data. Hash algorithms should have these properties—(1) changes in the given data should result in different hash values, and (2) computation of given data that has a particular hash value should be difficult.
- Encryption—a technique that creates apparently random data from the given plain text. Encryption algorithms should have the property that deriving the original data from the encrypted data should be difficult without the decryption key, and should be easy with the decryption key.
- Signature generation—a technique that combines hashing and encryption to create a signature which indicates an association between a given set of data and a particular entity, which is said to have signed the data. A digital signature technique should have the property that changes in the given text should result in changes in the signature, creation of given text resulting in a particular signature should be difficult, and creation of a particular digital signature without knowing the secret key of the original signer should be difficult.

It is useful from the DIG35 point of view to be able to show

1. that metadata has not been altered after some given time or version;
2. that the image data has not been altered after some given time or version;
3. that some metadata is properly associated with some image data.

Hashes or digital signatures of the metadata or image data can be used to show that the image pixel data or metadata have not been changed (1 and 2 above). A hash or digital signature of the combined (concatenated) hashes or signatures allows verification that the proper association between metadata and image pixel data. The technical specifications for these mechanisms are for further study.

Privacy of image pixel data or metadata is out of scope for DIG35.

NOTE: DIG35 welcomes comments in this area.

5.6 Metadata Persistence

Rules for metadata persistence are a fundamental component of a metadata systems design. If persistence be merely an issue of copying all metadata, then the implementation of a mechanism for metadata persistence is a manageable task.

It is not the case, however, that blindly copying metadata suffices. The complexity of metadata persistence is related to the set of reasonable expectations about the quality of the metadata following persistence. If the image is cropped, it is reasonable to expect that the Content Description metadata is updated so that it only describes objects relating to the image after cropping.

The following are a set of desirable properties for metadata. They are listed in increasing order of importance, and increasing difficulty of implementation

- Metadata should be well-formed, as defined by XML (or other encoding rules).
- Metadata should be structurally valid, as defined by an XML DTD (or other syntactic specification).
- Each metadatum, when considered in isolation from all other metadata, should have a value that is valid as defined by some specification. For instance, exposure times should be greater than zero.
- Metadata considered in pairs, triples, …, and n-tuples should be consistent as a group. For instance, exposure time and shutter speed should be pair-wise consistent.
- All metadata should be correct.

The Flashpix image file format suggested processing rules, applicable on an element by element basis, for metadata: In the Flashpix rules, some metadata should be copied, some should be deleted, and some should be copied only if the processing application understood the meaning of the metadata (and deleted otherwise).

The DIG35 approach for metadata persistence builds on the Flashpix effort, but with the variation that the processing application marks the metadata to indicate the processing that has taken place. The processing application might be creating, updating, deleting, or copying the metadata. The following rules cover each of those cases:

1. Metadata must be copied except where rule 2 or rule 3 applies.
2. Metadata must be deleted when the processing application "knows" the metadata should be deleted.
3. Metadata must be updated when the processing applications "know" the correct new value.
4. When metadata is copied the previous date stamp must be copied.
5. When metadata is created or updated, the associated date stamp must be set to "now".
6. When an application (with or without human intervention) has determined that metadata is correct, the date stamp of the metadata must be set to "now".

These metadata persistence rules permit a relatively simple test in subsequent processing applications. If a metadatum has a time stamp that is the same as the last image modification time, then there is no reason to doubt that the metadatum is incorrect. Metadata with time stamp(s) earlier than the last image modification time have not been identified as correct in the context of subsequent image and metadata modifications. Processing applications may suspect such metadata is incorrect.

Metadata correctness is oftentimes more relative and less absolute, however. The metadata in the file is in general the best available estimate of the true metadata. The following rules cover the use of the metadata by processing applications:

1. The processing applications must treat the metadata as reliable when the metadatum date stamp matches the last image modification time.
2. The processing application may attempt to determine consistency and validity when metadata date stamps are earlier than the last image modification time.
3. The processing application should treat the metadata as reliable when the only indication that metadata may be invalid or inconsistent is that the metadata date stamp is earlier than the last image modification time.

The metadata date stamps and the image modification time metadatum are discussed in the Annexes.

NOTE: It is left for further study to determine if processing applications need more information about the provenance of a particular metadatum. Processing applications could add attributes to the metadata elements to indicate that the processing application did or didn't understand the metadata, or that it created or modified the metadata, or that the metadatum was determined to be correct programmatically, or that a human operator deemed the metadata to be correct.

5.7 Intellectual Property Rights (IPR)

Intellectual Property Rights [IPR] is a general concept addressing the ownership and usage conditions of image content. Because the content of a file can be either created from scratch (original works) or built upon existing material (derivative works), the IPR may be different for each scenario. There are two basic categories of IPR, which are as follows: (1) Moral Rights (attached to the creator and non-transferable); and (2) Copyrights (conditional rights to copying, using, and exploiting content).

The World Intellectual Property Organization (WIPO) has stated that once IPR information is included in a file, then that information becomes an integral part of the file and must be conveyed without modification along with the file by all intermediate actors between creator and end user. This concept of conveying the IPR information is called “persistence.” Therefore, IPR data is important metadata, which is inherently associated with the image data.

Because IPR data cannot be erased or modified inside of the file (if it is to comply with legal requirements), it may become necessary either to add new data or to update the existing data. For instance, if an image is created by “collage” (i.e., compound image of multiple original images), then there should be a special set of metadata that give the provenance of all pieces so that the IPR of each image is respected. Another instance is when a photographer who is attached to one agency leaves the first agency and attaches to another and is allowed by the first agency to take all of his photographs with him. Similarly, if an agency disappears and a new agency is created, then the IPR metadata must persist and reflect this change.

The original IPR data cannot be erased, but additional data can be added. Another way to eliminate this type of situation is to use IPR data as a link to a secured database, which is managed by a Trusted Third Party (TTP), such as a law firm. IPR data is updated in the TTP log files, which can be easily accessed using the link, which is inside of the file. Such would be the case if the only IPR data that is present in the file is a watermark, inserted inside the image data. By detecting and extracting the value of the watermark, one gets the link to the TTP database, where he can read the latest, updated information. Access to IPR information, however, can be restricted in certain conditions, such as when the information is classified as “confidential.” Obviously, for the same reason, updating the IPR information, which could be done by adding information and a timestamp, is also restricted to conditional access.

All IPR metadata should be determined with a view to extensions, according to the future operation of automatic billing and crediting of both the end user and the creator. These systems called Electronic Copyright Management Systems (ECMS) and /or Intellectual Property Management and Protection (IPMP) are not completely defined as of today, although their main input and output data are those stated within this document.

ANNEXES

Annex A: Fundamental Metadata Types and Fields

A.1 Overview

The following diagram shows the relation of the Common Metadata and various components of DIG35 metadata definition.

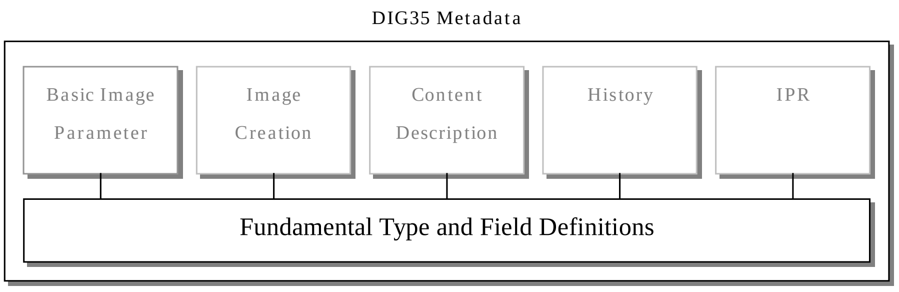


Figure A-1: High-level structure of the Fundamental Type and Field metadata

This annex defines the Fundamental Types and Fields that are common to all sub-blocks. The Types and Fields defined in this annex is intended only to be used or referred to in one of the other sub-blocks, and is not a block of data in itself.

All fields listed in this document are optional unless otherwise stated.

A DIG35 compliant metadata reader/editor must understand all fields unless otherwise stated. Note that a DIG35 metadata editor should not remove fields that are not understood when a DIG35 metadata file is modified.

A.2 Types

A.2.1 Atomic Data Types

The following atomic data types and characteristics are used in this specification.

Table A-1: Atomic Data Types and definitions

Atomic Data Types	Definitions
`Integer`	The string representation of a legal integer i.
`Non-negative Integer`	An `Integer` where `i` ≥ 0.
`Positive Integer`	An `Integer` where `i` > 0.
`Real`	The string representation of a legal real number `r`.
`Non-negative Real`	A `Real` where `r` ≥ 0.
`Rational`	The string representation of a legal rational number `r`.
`Non-negative Rational`	A `Rational` where `r` ≥ 0.
`String`	One or more characters.
`Boolean`	A string containing either “`true`” or “`false`".

A.2.2 Value Types

A.2.2.1 Identifier

An Identifier is a unique value. The "uniqueness" of the field could be only within a specific context, or a globally unique value. For example, the serial number of a camera is an identifier. It is unique only in the context of being combined with the manufacturer's name.

An element may contain a unique identifier that is unique within the context of the file.

A globally unique number (either supplied from a service, or privately generated) is another example of a unique identifier. Where identifiers are used within this specification, the context of their uniqueness is also defined. The Language Identifier is an example of an Identifier.

A.2.2.2 Reference

A Reference is a pointer to another object. The destination location is globally unique. An identifier may well be used to specify a unique location that a reference could point to.

For example, an identifier that is unique within a file could be used as the destination of a reference that specifies the filename (which could be "current file") and the identifier.

An URI is an example of a reference.

A.2.2.3 Enumerated Type

A Enumerated Type is a `String` that may only contain one of a number of known values. The `Boolean` type is an example of an enumerated value.

A.2.3 Defined Data Types

A.2.3.1 Date and Time

A.2.3.1.1 DateTime

The DateTime type contains a Date and a Time. All fields are optional. The DateTime type may contain Timestamp and Language Identifier fields.

A.2.3.1.2 Date

The Date type can be used as part of the DateTime type, or as a separate type. The Date type may contain Timestamp and Language Identifier fields. The Date type contains a year, a month and a day. It could also contain a season and a comment. All fields are optional.

Year: This field is an `Integer`. Positive values used for AD and negative values for BC. Where a year is not specified, it does not have a default value. Rather – the year is not defined.

Month: This field is a `Positive Integer`. Valid values are from 1 to 12 inclusive.

Table A-2: Month-Value pair

Value	Month	Value	Month	Value	Month
1	January	5	May	9	September
2	February	6	June	10	October
3	March	7	July	11	November
4	April	8	August	12	December

Where a month is not specified, the day is treated as a year day. Where neither the day nor month is specified, then the month is not defined.

Day: This field is a `Positive Integer`. It can either be used as the day of the month, or the day of the year. The day must be between 1 and 366 inclusive. If a day is specified, and not a month, then the day is treated as the day of the year.

Season: This field is a text description of a season. This field is a `String` and can contain a Language Identifier. Examples include: “Spring”, “Summer”, “Autumn”, and “Winter.”

Comment: See Comment for more information on this field.

A.2.3.1.3 Time

The Time type can be used as part of the DateTime type, or as a separate type. The Time type may contain Timestamp and Language Identifier fields.

A Time contains an hour, a minute and a second. The time can also contain a comment. The time field can be used to either specify the **time of day**, or an **elapsed time**. All fields are optional.

Where a time specifies a time of day, the total time specified is less than 24 hours. Time components are added together to calculate the total time. For example

10 hours 82 minutes = 11 hours 22 minutes

Either encoding is suitable and both are equivalent.

Hour: This field is a `Non-negative Integer`.

Minute: This field is a `Non-negative Integer`.

Second: This field is a `Non-negative Real`. Fractional seconds can be specified.

Comment: See Comment for more information on this field.
Examples could include “Morning”, “Just after lunch.”

A.2.3.2 Address

The Address type is used to define the street address of an object or location. For example, it may be used to describe the address an image was captured, or the address of the intellectual property owner of an image.

The Address type may contain Timestamp and Language Identifier fields.

Name: It is a descriptive field for the address. An example would be “Yankee Stadium”.

This field is a `String`. The Name field may contain a Language Identifier field.

Address: There are one or more address fields. Multiple fields are used to specify the complete address. Addresses from different localities are specified using various means.

This field is a `String`. The Address field may contain a Language Identifier field.

Type: This is the name of this part of the address. Examples include "street" or "state". This field is a `String`. ISO 3166-2 specifies country subdivisions and the types of these divisions. These subdivision types could optionally be used to specify the address type.

Value: This is the value of this part of the address. Where the type is a state, this field contains the name of the state. Where the type is a street, this field contains the name of the street. This field is a `String`. ISO 3166-2 lists country subdivision codes. These codes can optionally be used in this field, when the field is being used to specify a country subdivision.

Examples:

```
Type="Unit"       Value="10"
Type="Street"     Value="23 Fleet Street"
Type="City"       Value="Carlingford"
Type="State"      Value="New South Wales"
```

Postcode / Zipcode: This field is a `String`. It contains the postcode (or zip code) of the address. This field should not be limited in length.

The field has the title “Postcode” or “Zipcode”. Only one of the fields may exist. For example, an address cannot contain a postcode and a zipcode.

Country: This field is either a `String` or an `Identifier`. It contains the country of the address. The field can either contain the country code as defined in [ISO 3166-1] or a string identifying the country. The ISO 3166-1 country code is preferred. . The Country field may contain a Language Identifier field.

Type: This field defines the type of the address. The address type would include whether the address is a home address or a business address. The field is a `String`.

A.2.3.3 Phone Number

The Phone Number type is used to describe a phone number. The Phone Number type may contain a Timestamp field.

A.2.3.3.1 International Phone Code

This is a `Positive Integer` that contains the country code part of a phone number.

A.2.3.3.2 Local Phone Area Code

This is a `Positive Integer` that contains the local area code part of a phone number.

A.2.3.3.3 Phone Number

This is a `Positive Integer` that contains the local phone number.

A.2.3.3.4 Extension

This is a `Positive Integer` that contains the extension part of the phone number.

A.2.3.3.5 Type

This field defines the type of the phone number. The phone number type would include whether the address is a home address or a business address. The field is a `String`.

Example:

```
Phone Number:(+61) 2 9212 2646

<COUNTY_CODE>61</COUNTRY_CODE>
<AREA>2</AREA>
<LOCAL>92122646</LOCAL>
```

A.2.3.4 Person Description

The Person type is used to describe a person. The sub-fields are compatible with the vCard description [RFC 2426].

The Person type may contain Timestamp and Language Identifier.

A.2.3.4.1 Name Title

The field is a `String`. It contains the person's title.

Examples:

- Duke of York
- Queen of England
- President of Australia

The Name Title field may contain Timestamp and Language Identifier.

A.2.3.4.2 Name

This section defines a framework to describe a person's name. A person's name is composed of multiple name components (e.g. given name(s) and family name(s)). ***The order of the name component fields specifies the full name of the person. For example, in languages where the family name is usually placed before the given name, then they would appear in this order in the file.***

A Person Description may contain zero or name fields.

Name Component: This field contains a single portion (word) of the name of a person. A name component field can contain a single initial rather than a complete word. To specify the full name of a person, multiple name component fields are used.

The Name Component field may contain Timestamp and Language Identifier.

> **Type:** This field defines the type of the Name Component. This field would include whether the name component is a Suffix, Prefix, Given or Family name. The field is an `Identifier`. Valid values are:
> - Prefix
> - Given
> - Family
> - Suffix

Example:

```
Name: Rev Dr Chuck Berry III Esq

<PERSON_NAME>
  <NC TYPE="prefix">Rev</NC>
  <NC TYPE="prefix">Dr</NC>
  <NC TYPE="given">Chuck</NC>
  <NC TYPE="family">Berry</NC>
  <NC TYPE="suffix">III</NC>
  <NC TYPE="suffix">Esq</NC>
</PERSON_NAME>

Name: 石井 克己

<PERSON_NAME xml:lang="ja">
  <NC TYPE="family">石井</NC>
  <NC TYPE="given">克己</NC>
</PERSON_NAME>
<PERSON_NAME xml:lang="en">
  <NC TYPE="given">Katsuki</NC>
  <NC TYPE="family">Ishii</NC>
</PERSON_NAME>
```

A.2.3.4.3 Nick Name

It contains a nick name of the person. E.g. “Jimmy.” The field is a `String`.

The Nick Name field may contain Timestamp and Language Identifier.

A.2.3.4.4 Job Title

The field is a `String`. It contains the person’s job title.

Examples:

- Engineer
- Brain Surgeon
- Racing Car Driver

The Job Title field may contain Timestamp and Language Identifier.

A.2.3.4.5 Company

This field is a `String`. It contains the company for which a person works.

The Company field may contain Timestamp and Language Identifier.

A.2.3.4.6 Address

This field contains address information for the person. For example, it can contain a home address or a work address. It does not necessarily contain the address depicted within the image, but instead information about the person.

See Address for the format of this field.

A.2.3.4.7 Phone Number

This field contains phone number information.

See Phone Number for the format of this field.

A.2.3.4.8 Email Address

Email: This field is a `String`. It contains the email address of the person.

Type: This field is a `String`. It contains the type of the email address.

Examples:

- Home
- Business

A.2.3.4.9 Web Page

Web Page: This field is a `String`. It contains the home web page of the person.

Type: This field is a `String`. It contains the type of the web page.

Examples:

- Home
- Business

A.2.3.4.10 Date Of Birth

See DateTime for the format of this field. It contains the birth date of the person.

A.2.3.4.11 Age

The age of a person can be included.

Examples:

- 10 years
- 2 years, 3 months
- 6 months
- 16 days

See DateTime for the format of this field.

A.2.3.4.12 Comment

See Comment for more information on this field.

A.2.3.5 Organization Description

The Organization type is used to describe an organization. The sub-fields are compatible with the vCard description [RFC 2426]. The Organization type may contain Timestamp and Language Identifier.

A.2.3.5.1 Name

This field is a `String`. It contains the name of the organization.

A.2.3.5.2 Address

This field contains address information for the organization. It does not necessarily contain the address depicted within the image, but instead information about the organization.

See Address for the format of this field.

A.2.3.5.3 Phone Number

This field contains phone number information.

See Phone Number for the format of this field.

A.2.3.5.4 Email Address

Email: This field is a `String`. It contains the email address of the organization.

Type: This field is a `String`. It contains the type of the email address.

Examples:

- Feedback
- Web Master

A.2.3.5.5 Web Page

Web Page: This field is a `String`. It contains the home web page of the person.

Type: This field is a `String`. It contains the type of the web page.

Examples:

- Homepage
- Company Information Page

A.2.3.5.6 Comment

See Comment for more information on this field.

A.2.3.6 Location

The Location type is used to describe the physical location of an object or a scene. For example, it may be used to describe an object within an image, or the location of a camera at the time of capture.

The Location is the physical location, whereas the Position is the position of an object relative to the picture.

Examples of locations:
Under the table
Longitude: 10 degrees, Latitude: 20 degrees

Example of positions:
Top left corner
Rectangle: x=0.1, y=0.1, width=0.2, height=0.3

The Location type may contain Timestamp and Language Identifier fields.

A.2.3.6.1 Coordinate Location

The coordinate location is used to describe the location (altitude / longitude / latitude) of an object. It is used to describe the content of an image along with the location of a camera.

While the coordinate location may have come from a GPS (and a GPS block may or may not be present in the metadata), the values in the coordinate location may have come for some other means. For this reason, the location information is a more general system for storing the location than the GPS system. The location information and the raw GPS data are stored in different formats.

The meridian through Greenwich (Great Britain) is defined with the value longitude l =0. The longitude l of a point P on the surface is the angle between the planes through its meridian and the Greenwich meridian. The longitude is counted from Greenwich up to l =±180° in east(+) and west(-) directions.

The latitude j of a point P is the angle between a line normal to its parallel and the equatorial plane (j =0). On a sphere this normal line will be the connecting line between its center and the point P. On the elliptical earth this line will only pass the center if P is situated at the equator. The latitude is counted from the equator up to j =±90° in north (+) and south (-) directions.

Longitude: This field contains the longitude, represented in decimal degrees and fractions of degrees. The value of this field is a `Real`.

Examples:

```
 138.700
-122.450
```

Latitude: This field contains the latitude, represented in decimal degrees and fractions of degrees. The value of this field is a `Real`

Examples:

```
35.383
37.767
```

The Coordinate Location type may contain a Timestamp field.

> Note: There is the option of storing the coordinate location using degrees, minutes and seconds for longitude and latitude. It is yet to be determined which system is the most useful. DIG35 welcome comments on this issue.

A.2.3.6.2 UTM Coordinate Location

The section defines a coordinate location using UTM (Universal Transverse Mercator). The UTM coordinate grid divides the world into 60 zones of latitude, each 6 degrees wide, that extend from 84 degrees north to 80 degrees south. These zones begin at 180 degrees longitude and are numbered consecutively eastward. In the northern hemisphere each zone's northing coordinate begin at the equator as 0,000,000 and are numbered north in meters. For the southern hemisphere, the equator is labeled 10,000,000. The easting coordinates are measured from an artificial reference line drawn parallel and 500,000 meters to the west of the zone's central meridian. (Thus each central meridian is numbered 500,000.) In the case of Zone 13, the central meridian is 105 degrees west.

The UTM Coordinate Location type may contain a Timestamp field.

Altitude: This field would contain the distance in meters. Zero is sea level, positive is above, and negative is below. The value of this field is a `Real`.

Map Datum: The Map Datum field contains a `String` that indicates the geodetic survey data. Geodetic datums define the reference systems that describe the size and shape of the earth. If no map datum is specified, "WSG-84" is assumed.

Zone: This figure is a `Non-negative Integer` and specifies the UTM zone. It contains the UTM zone which is a number between 1 and 60. UTM zones are described as:

01	177W	180W-174W	31	003E	000E-006E
02	171W	174W-168W	32	009E	006E-012E
03	165W	168W-162W	33	015E	012E-018E
04	159W	162W-156W	34	021E	018E-024E
05	153W	156W-150W	35	027E	024E-030E
06	147W	150W-144W	36	033E	030E-036E
07	141W	144W-138W	37	039E	036E-042E
08	135W	138W-132W	38	045E	042E-048E
09	129W	132W-126W	39	051E	048E-054E
10	123W	126W-120W	40	057E	054E-060E
11	117W	120W-114W	41	063E	060E-066E
12	111W	114W-108W	42	069E	066E-072E
13	105W	108W-102W	43	075E	072E-078E
14	099W	102W-096W	44	081E	078E-084E
15	093W	096W-090W	45	087E	084E-090E
16	087W	090W-084W	46	093E	090E-096E
17	081W	084W-078W	47	099E	096E-102E
18	075W	078W-072W	48	105E	102E-108E
19	069W	072W-066W	49	111E	108E-114E
20	063W	066W-060W	50	117E	114E-120E
21	057W	060W-054W	51	123E	120E-126E
22	051W	054W-048W	52	129E	126E-132E
23	045W	048W-042W	53	135E	132E-138E
24	039W	042W-036W	54	141E	138E-144E
25	033W	036W-030W	55	147E	144E-150E
26	027W	030W-024W	56	153E	150E-162E
27	021W	024W-018W	57	159E	156E-162E
28	015W	018W-012W	58	165E	162E-168E
29	009W	012W-006W	59	171E	168E-174E
30	003W	006W-000E	60	177E	174E-180W

Longitude: This field would include a distance in UTM. The value of this field is a `Non-negative Real`.

Examples:

```
704250
317456
```

Latitude: This fielde would include a distance in UTM. The value of this field is a `Non-negative Integer`

Examples:

```
      3391520
      6260251
```

A.2.3.6.3 Address

This section defines the location of an object using a street address. See Address for the format of this field.

A.2.3.6.4 Comment

See Comment for more information on this field.

Example: "Under the table"

A.2.3.6.5 Raw GPS Information

Note: The fields in this section are taken from Exif and TIFF/EP. The coordinate system units, however, are in UTM.

[Editor's Note: To be completed]

If GPS information is specified, then if information for Coordinate Location (latitude, longitude and altitude) is present, the matching fields in the metadata must be filled in.

GPS Version ID: This field encodes the version of this GPS Information section as a four tier revision number, for example 2.0.0.0, as a `String`. This revision number has the form of w.x.y.z where w=0-255, x=0-255, y=0-255, and z=0-255. The present version is 2.0.0.0, to maintain compatibility with the Exif 2 specification.

GPS Zone: This field contains the UTM Zone the device is located within in UTM coordinates. The field is a `Non-negative Real`.

GPS Longitude: This field contains the device's longitude in UTM coordinates. The field is a `Non-negative Real`.

GPS Latitude: This field contains the camera's latitude in UTM coordinates. The field is a `Non-negative Real`.

GPS Altitude Ref: The ***GPS Altitude Ref*** tag value shall contain a one BYTE value of "0", indicating sea level, for the current version. The altitude reading is given in meters relative to sea level.

GPS Altitude: This value contains a single `Real` value equal to the camera's altitude in meters. Negative specifies that the altitude is below sea level.

GPS Time Stamp: The ***GPS Time Stamp*** tag contains a DateTime specifying the time given by the GPS clock when the GPS location was calculated.

GPS Satellites: The ***GPS Satellites*** tag contains a `String` that lists the satellites used to determine the camera position. This tag can be used to describe the number of satellites, their ID number, angle of elevation, azimuth, SNR and other information. The format is not specified. If the GPS receiver is incapable of taking measurements, this field can be omitted.

GPS Status: The ***GPS Status*** tag contains an Identifier, where A means the measurement is in progress, and V means the measurement is interrupted.

GPS Measure Mode: The ***GPS Measure Mode*** tag contains an Identifier, where 2 indicates a two-dimensional measurement and 3 indicates a three-dimensional measurement is in progress.

GPS DOP: The ***GPS DOP*** tag contains a `Non-negative Real` value indicating the GPS DOP (data degree of precision). An HDOP value is written during a two-dimensional measurement, and a PDOP value is written during a three-dimensional measurement.

GPS Speed Ref: This field contains a null-terminated ASCII character, where K means Kilometers per hour, M means Miles per hour, and N means Knots.

[Editor's note: The speed is to be in m/s and thus this field is not required.]

GPS Speed: This field contains a `Non-negative Real` value indicating the speed of the GPS receiver. The value is in meters per second.

GPS Track Ref: This field contains a null-terminated ASCII character indicating the reference for the direction of the GPS receiver. T means true direction and M means magnetic direction.

GPS Track: This field contains a `Non-negative Real` value indicating the direction of the GPS receiver movement in degrees. 0 indicates North and 90 indicate East.

GPS Image Dir Ref: This field contains a null-terminated ASCII character indicating the reference for the direction of the image when it is captured. T means true direction and M means magnetic direction.

GPS Image Direction: This field contains a `Non-negative Real` value indicating the direction of the image when it was captured in degrees. 0 indicates North and 90 indicate East.

GPS Map Datum: This field contains a `String` indicates the geodetic survey data used by the GPS receiver. If the survey data is restricted to Japan, the value of this tag is “TOKYO” or “WSG-84”.

GPS Dest Zone: This field contains the UTM Zone for the GPS destination point in UTM coordinates. The field is a `Non-negative Real`.

GPS Dest Longitude: This field contains the GPS destination point longitude in UTM coordinates. The field is a `Non-negative Real`.

GPS Dest Latitude: This field contains the GPS destination point latitude in UTM coordinates. The field is a `Non-negative Real`.

GPS Dest Bearing Ref: This filed contains a null-terminated ASCII character indicating the reference for the bearing to the destination point. T means true direction and M means magnetic direction.

GPS Dest Bearing: The filed contains a `Non-negative Real` value indicating the bearing to the destination point in degrees. 0 indicates North and 90 indicate East.

GPS Dest Distance Ref: This field contains a null-terminated ASCII character, where K means Kilometers, M means Miles, and N means Knots.

GPS Dest Distance: The field contains a `Non-negative Real` value indicating the distance to the destination point. The distance is in meters.

A.2.3.7 Direction

The Direction type specifies a three-dimensional heading. While this type is primarily used to specify the direction a camera is facing, it could also be used to specify information about an object in a scientific photograph for example. When calculating the direction the camera is facing, first the yaw is applied, then the pitch, then the roll.

The fields in this section have been taken from [John Denker]. The Direction type can contain a Timestamp.

A.2.3.7.1 Yaw

This field is the direction the capture device is facing. The field is a `Non-negative Real` value between 0 and 360, and is measured in degrees. North is 0, East is 90, South 180 and West is 270.

A.2.3.7.2 Pitch

This field is a measure of the elevation angle of the capture device. This field is a `Real` value between –90 and +90, also measured in degrees. 0 facing horizontal. 90 is facing vertically straight upwards, and –90 vertically downwards.

A.2.3.7.3 Roll

This field is a measure of the rotation angle of the capture device. This field is a `Real` value between –180 and 180, also measured in degrees. 0 facing horizontal. 90 where the device is rotated clockwise and the left of the device is facing upwards, and –90 where the device is rotated anti-clockwise. 180 is upside down.

A.2.3.8 Position

Note: The coordinate system is not normalised. Also – the coordinate system is in pixels. Values between 0 and 1 have been considered. DIG35 welcomes comments on this issue.

The Position type is used to specify the position of an object, within an image. The Position can be one of the following:

- An x, y single point.
- A rectangular area (specified as an x, y, width and height)
- A set of splines that represent an area of the image.
- A free-text comment field

Coordinates are stored in pixels, where 0, 0 being the top left of the image, W (image width in pixels) on the top right, and H (image height in pixels) being the bottom left. (See Figure A-2)

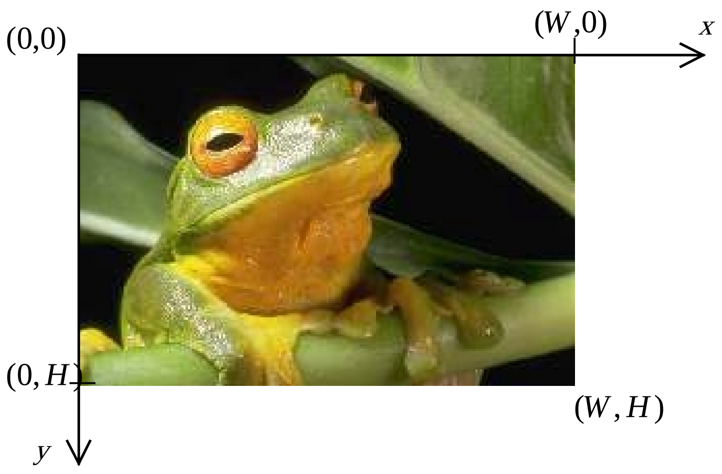


Figure A-2: Coordinate system

Note that this information may become useless if the image is cropped or manipulated so that the image's aspect ratio changes.

The Position type may contain a Timestamp.

See Location for the difference between the Position and Location types

A.2.3.8.1 Single Point

This element contains two fields:

- x – Non-negative Integer value
- y – Non-negative Integer value

A.2.3.8.2 Rectangular Region

This element contains four fields:

- x – Non-negative Integer value (The left of the rectangle).
- y – Non-negative Integer value (The bottom of the rectangle).
- width – Non-negative Integer value
- height – Non-negative Integer value

A.2.3.8.3 Arbitrary Region

This element contains one or more splines. Each spline can contain either 2 or 6 coordinates.

- x1, y1 – mandatory control point (Non-negative Integer values)
- x2, y2, x3, y3 – optional control points for spline (Non-negative Integer values)

A straight line is defined if only two coordinates are specified.

Where an arbitrary region is specified, a rectangular region Rectangular Region must also be specified (which is the bounding box of the arbitrary region). A standard DIG35 compliant metadata reader / editor has the option of understanding arbitrary regions. Note that rectangular regions Rectangular Region must be understood by all DIG35 metadata compliant readers.

A.2.3.8.4 Comment

This field can describe the position of an object less accurately than one of the above methods. For example, this field may contain “Bottom left-hand corner” or “Second from the left in the top row”.

See Comment for more information on this field.

A.2.3.9 Product Details

This section specifies details about a product (hardware or software). This element may contain a Timestamp and a Language Identifier.

A.2.3.9.1 Manufacturer Name

This element specifies the name of the manufacturer or vendor of the product. The value of this element is encoded as a `String`.

A.2.3.9.2 Model Name

This element specifies the model name or number of the product. The value of this element is encoded as a `String`.

A.2.3.9.3 Version Number

This element specifies the version number of the product. The value of this element is encoded as a `String`.

A.3 Fields

A.3.1 Language Identifier

The field is formatted according to [RFC1766]. When a metadata field contains a Language Identifier, it specifies the language in which the metadata is stored. See Enumerated Type for the format of this field.

A.3.2 Timestamp

When a metadata field contains a Timestamp, it specifies the time that the metadata was generated.

A Timestamp differs from a DateTime in two ways:

- A Timestamp specifies all date and time fields, whereas a DateTime may specify just a month or a season.
- A Timestamp is used to specify when a metadata field was generated, whereas a DateTime is used to specify information such as a date of birth, or the date a picture was taken.

Year: Positive values used for AD and negative values for BC. This field is an `Integer`. The Year field is mandatory.

Month: Valid values are from 1 to 12 inclusive. This field is a `Positive Integer`. See Table A-2 for the month and value pair. The Month field is mandatory.

Day: It is the day of the month. The Day must be between 1 and 31 inclusive. This field is a `Positive Integer`. The Day field is mandatory.

Hour: The field is between 0 and 23. This field is a `Non-negative Integer`. The Hour field is mandatory.

Minute: The field must contain a value between 0 and 59. This field is a `Non-negative Integer`. The Minute field is mandatory.

Second: The field must contain a value equal to or greater than 0 and strictly less than 60. Fractional seconds can be specified. This field is a `Non-negative Real`. The seconds field is optional.

A.3.3 Comment

The Comment field is used to specify extra information to the field it contains that cannot be described otherwise within the metadata.

See String for the format of this field. The Comment field may contain Timestamp and Language Identifier.

[Editor's note: This field should be not used when the metadata can be described by other fields. E.g. storing an address in a comment field rather than an address field defeats the purpose of a defined metadata model.]

Annex B: Basic Image Parameter Metadata

B.1 Overview

This annex defines basic image parameter metadata that contains generic information about the image, such as the image size and number of components. While these metadata that are commonly referred to as the "header" of an image file format, the scope of this annex is to define metadata that are file format independent. Thus this metadata should be considered informational and not to be used to decode the image data stored in the associated file.

B.2 Structure

The following diagram illustrates the logical structure of the Basic Image Parameter Metadata.

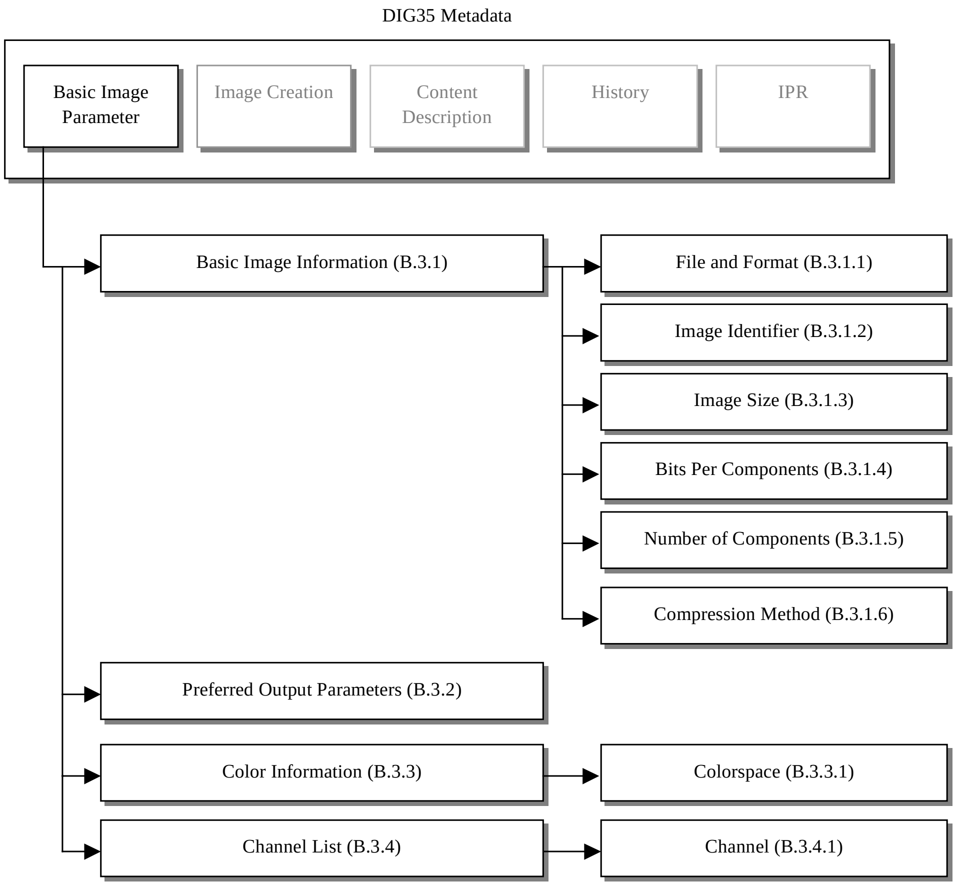

DIG35 Metadata
Basic Image Parameter
Image Creation
Content Description
History
IPR
Basic Image Information (B.3.1)
File and Format (B.3.1.1)
Image Identifier (B.3.1.2)
Image Size (B.3.1.3)
Bits Per Components (B.3.1.4)
Number of Components (B.3.1.5)
Compression Method (B.3.1.6)
Preferred Output Parameters (B.3.2)
Color Information (B.3.3)
Colorspace (B.3.3.1)
Channel List (B.3.4)
Channel (B.3.4.1)

Figure B-1: Basic Image Parameter metadata structure

B.3 Definition

B.3.1 Basic Image Information

This section defines generic information about the image, such as the image size and number of components. This element can contain a Timestamp and a Language Identifier.

B.3.1.1 File and Format

This specifies the image file and format that the metadata is associated with. This element can contain a Timestamp.

File Name: This field specifies the name of image file. The value may be either a `String`, or a `Reference`.

File Format Name: This field specifies the name of the image file format. For example, JIFF/JPEG, Flashpix, Exif and TIFF. This value must be a `String`.

Version: This field specifies the version of the file format. The value must be a `String`.

B.3.1.2 Image Identifier

This field specifies a image identifier that must uniquely identify the image(s) which bear them. The format may be globally unique (e.g. UUID), vendor or application dependent. The value is a `String`. This element can contain a Timestamp and a Language Identifier.

B.3.1.3 Image Size

This specifies the size of the image. For multiple-resolution image file formats, it shall specify the highest resolution. It has two values. This element can contain a Timestamp.

Width: This specifies the width of the image, in pixels. The value of this field must be a `Positive Integer`.

Height: This specifies the height of the image, in pixels. The value of this field must be a `Positive Integer`.

B.3.1.4 Bits per Components

This specifies the bit depth of each component. The order of bit depth values should be same as the actual order those components are enumerated within the image data. Each bit depth component value must be a `Positive Integer`. Where the number of bits per component is the same for all components, a single value may be specified. This field can contain a Timestamp.

B.3.1.5 Number of Components

This parameter specifies the number of components in the image. This value must be a `Positive Integer`. This field can contain a Timestamp.

B.3.1.6 Compression Method

This specified the compression method used to store the image data. Values include JPEG, JPEG2000, etc. This value must be a `String`.

B.3.2 Preferred Output Parameter

This specifies the preferred height and width, respectively, for a particular output device. This field can contain a Timestamp.

Width: This field specifies the preferred width of the image in meters. The value must be a `Positive Integer`.

Height: This field specifies the preferred height of the image in meters. The value must be `Positive Integer`.

B.3.3 Color Information

This section specifies the colorspace of the decompressed image data. Note that internal to the compression process, the image coder may convert the data to a different colorspace. However, this process is considered a black box at the file format level, and that internal colorspace is not exposed to the end user. This field is recommended. This field can contain a Timestamp and a Language Identifier.

B.3.3.1 Colorspace

This specifies the colorspace of the decompressed image data as an ICC profile. The profile may be specified either by a profile name or an URL. If it is Named, then the profile is known by a well-defined name and thus is not included in the file; the name of the desired profile should be specified. If the value is `URL`, then the desired profile is located at the `URL`.

Profile Name: This field specifies the well-known name of the ICC profile that specifies the colorspace of the decompressed image data. If the value of the `profile` field is not `Named`, then this field must not be found.

Profile URL: This field specifies the location of the ICC profile that specifies the colorspace of the decompressed image data. If the value of the profile field is not URL, then this field must not be found.

B.3.4 Channel Information

This field contains the names and bit-depths of the individual channels of the image. For each channel in the image, this field contains an Channel type which specifies the information for one particular channel. The order of the `Channels` in the `Channel Information` field must be the same as the order of the channels in the compressed image data.

Number of Channels: This field specifies the number of channels in the image. The value of this field must be the same as the number of `Channels` contained in the `Channel Information`. The field value must be a `Positive Integer`.

Premultiplied: This field specifies whether the opacity channel of the image has been premultiplied into the color channels. Legal values of this field is `Boolean`. If the value of this field is `true`, then the opacity channel has been premultiplied into all of the other channels. If the value of this field is `false`, then the opacity channel has not been premultiplied into any channels. This field is optional. If it is not present, the default value is `false`

B.3.4.1 Channel

This field specifies the name and bit-depth of a single channel from the image.

Name: This field specifies the name of the channel. For red, green, blue, cyan, magenta, yellow, black or opacity channels, the value of this field must be `R`, `G`, `B`, `C`, `M`, `Y`, `K` or `A`, respectively. For other channel types, such as are often found in medical or multi-spectral images, it is up to the application to standardize on a set of channel names.

Size: This field specifies the bit-depth of the channel. The value must be a `Positive Integer`.

Annex C: Image Creation Metadata

C.1 Overview

This annex defines metadata that are related to the creation of a digital image. The scope of this annex is applicable to metadata fields that are relevant to the creation of the digital image data, i.e. camera and scanner device information and its capture condition as well as the software or firmware to create such image. It defines the "how" metadata that specifies the pedigree of the image.

C.2 Structure

The following diagram illustrates the logical structure of the Image Creation metadata.

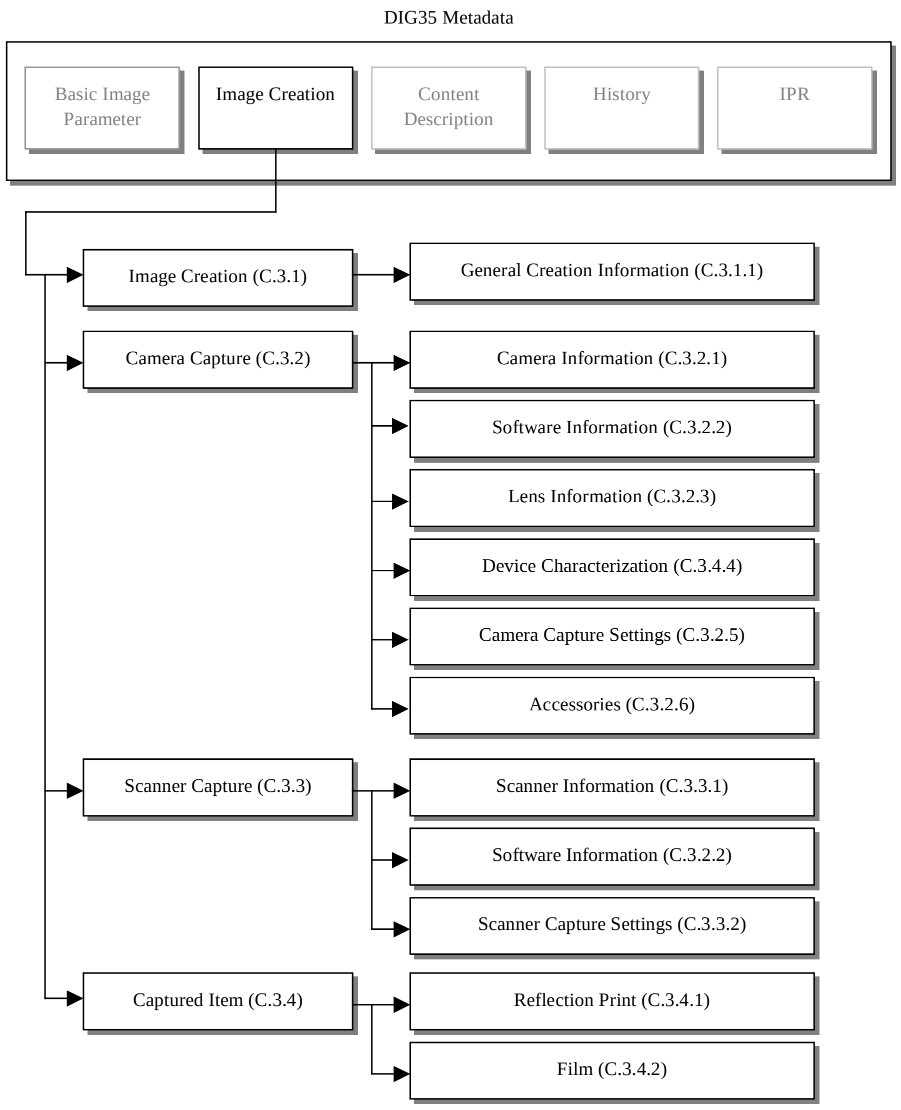

DIG35 Metadata
Basic Image Parameter
Image Creation
Content Description
History
IPR
Image Creation (C.3.1)
General Creation Information (C.3.1.1)
Camera Capture (C.3.2)
Camera Information (C.3.2.1)
Software Information (C.3.2.2)
Lens Information (C.3.2.3)
Device Characterization (C.3.4.4)
Camera Capture Settings (C.3.2.5)
Accessories (C.3.2.6)
Scanner Capture (C.3.3)
Scanner Information (C.3.3.1)
Software Information (C.3.2.2)
Scanner Capture Settings (C.3.3.2)
Captured Item (C.3.4)
Reflection Print (C.3.4.1)
Film (C.3.4.2)

Figure C-1: Image Creation metadata structure

C.3 Definition

C.3.1 Image Creation Metadata

C.3.1.1 General Creation Information

These fields specify general information on how the image was created. Applications may choose to skip further parsing based on the values stored here. For example if the application is only interested in Camera metadata, it can skip additional parsing based on the Image source value. Each field can contain a Timestamp and a Language Identifier.

Capture Time: This field specifies the date and time the image was captured. This field should be stored when the capture process started. (E.g. it may be a 8 minute exposure.) This field should never be changed after it is written in the image capture device. The value of this field must be a DateTime.

Image Source: This field specifies the device source of the digital file, such as a film scanner, reflection print scanner, or digital camera. The value of this field is encoded as a `String`. Possible values are, “Film scanner”, “Reflection print scanner”, “Digital camera”, “Still from video” or “Computer graphics.”

Scene Type: This field specifies the type of scene that was captured. It differentiates “original scenes” (direct capture of real-world scenes) from “second generation scenes” (images captured from pre-existing hardcopy images). It provides further differentiation for scenes that are digitally composed. The value of this field is encoded as a `String`. Possible values are “Original scene”, “Second generation scene” or “Digital scene generation.”

Image Creator: This field specifies the name of the image creator. The image creator could be, for example, the photographer who captured the original picture on film, the illustrator, or graphic artist who conducted the image-creation process, etc. The value of this field is a `String`.

Operator Organization: This field specifies the name of the service bureau, photofinisher, or organization where the image capture process (photographed, scanned or created by software) is conducted. See organization for the format of this field.

Operator ID: This field specifies a name or ID for the person conducting the capture process. The value of this field is encoded as an `Identifier`.

C.3.2 Camera Capture Metadata

This field specifies a camera capture of a scene. It optionally contains camera information, lens information, device characterization and camera capture settings.

C.3.2.1 Camera Information

This section specifies information about the camera that captured the scene. It is recommended that applications are able to create a unique value of the camera by combining all fields.

See Product Description for the format of this field.

C.3.2.2 Software Information

This section specifies information about the software or firmware used to capture the image.

See Product Description for the format of this field.

C.3.2.3 Lens Information

This section specifies information about the lens that captured the scene. It is recommended that applications are able to create a unique value of the lens by combining all fields.

See Product Description for the format of this field.

C.3.2.4 Device Characterization

This section specifies the technical characterization of the digital capture device.

Sensor Technology: This field specifies either the type of image sensor or the sensing method used in the camera or image-capturing device. The value of this field is encoded as a `String`. Valid values include; "One-chip color area", "Two-chip color area", "Three-chip color area", "Color sequential area", "Trilinear" and "Color sequential linear sensor." However, values are device specific.

Focal Plane Resolution: This field specifies the number of pixels in the X and Y directions for the main image respectively. They specify the actual focal plane X and Y resolutions at the focal plane of the camera. These values must be valid `Non-negative Integer`..

Spectral Sensitivity: This field can be used to describe the spectral sensitivity of each channel of the camera used to capture the image. It is useful for certain scientific applications. The `String` is compatible with [ASMT E1708-95].

ISO Saturation Speed Rating: This field specifies the ISO saturation speed rating classification as defined in [ISO 12232]. The value of this field is encoded as a `Real`.

ISO Noise Speed Rating: This field specifies the ISO noise-based speed rating classification as defined in [ISO 12232]. The value of this field is encoded as a `Real`.

Spatial Frequency Response: This specifies the spatial frequency response (SFR) of the image capturing device. The device measured SFR data, described in [ISO 12233], can be stored as a table of spatial frequencies, horizontal SFR values, vertical SFR values, and diagonal SFR values. The following is a simple example of measured SFR data table.

Table C-1: Sample Frequency Response

Spatial frequency (lw/ph [i])	Horizontal SFR	Vertical SFR
0.1	1.00	1.00
0.2	0.90	0.95
0.3	0.80	0.85

CFA Pattern: Encodes the actual color filter array (CFA) geometric pattern of the image sensor used to capture a single-sensor color image. It is not relevant for all sensing methods. The data contains the minimum number of rows and columns of filter color values that uniquely specify the color filter array. The color filters are "`Red`", "`Green`", "`Blue`", "`Cyan`", "`Magenta`", "`Yellow`" and "`White.`" The following table shows a simple example of a CFA Pattern, surrounded in bold lines, in a color filter array.

Table C-2: Sample CFA Pattern

Green	Red	Green	Red	…
Blue	Green	Blue	Green	…
Green	Red	Green	Red	…
Blue	Green	Blue	Green	…
…	…	…	…	…

OECF: This field specifies the opto-electronic conversion function (OECF). The OECF is the relationship between the optical input and the image file code value outputs of an electronic camera. The property allows OECF values defined in [ISO 14524] to be stored as a table of values. The following table shows a simple example of measured OECF data.

Table C-3: An example of measured OECF data

Log exposure	Red output level	**Green output level**	Blue output level
-3.0	10.2	12.5	8.9
-2.0	48.1	47.5	48.3
-1.0	150.2	152.0	149.8

Maximum Aperture: This field specifies the maximum possible aperture opening (minimum lens f-number) of the camera or image capturing device, using APEX units. The value of this field must be a `Non-negative Real`.

[i] line widths per picture height

C.3.2.5 Camera Capture Settings

This section describes the camera settings used when the image was captured. New generations of digital and film cameras make it possible to capture more information about the conditions under which a picture was taken. This may include information about the lens aperture and exposure time, whether a flash was used, which lens was used, etc. This technical information is useful to professional and serious amateur photographers. In addition, some of these properties are useful to image database applications for populating values useful to advanced imaging applications and algorithms as well as image analysis and retrieval.

Exposure Time: This field specifies the exposure time used when the image was captured. The value of this field must be either a `Non-negative Real` or a `Non-negative Rational` and the units in seconds.

The following table shows example values used as camera information.

Table C-4: Typical Exposure Time and corresponding Shutter Speed values

Exposure Time	15	8	4	2	1	1/2	1/4	1/8
Shutter Speed Value	-4	-3	-2	-1	0	1	2	3

Exposure Time	1/15	1/30	1/60	1/125	1/250	1/500	1/1000	1/2000
Shutter Speed Value	4	5	6	7	8	9	10	11

Note that Exposure Times and Shutter Speed Values have the following relation.

Shutter Speed Value (Tv) = $-\log_2$ (Exposure Time) (1)

See also: Shutter Speed Value

Shutter Speed Value: This field specifies the shutter speed value used when the image was captured. The value of this field must be a `Real` and the units are APEX (Additive System of Photographic Exposure) values.

See also: Exposure Time

F-Number: This field specifies the lens f-number (ratio of lens aperture to focal length) used when the image was captured. The value of this field must be a `Non-negative Real`.

The following table shows example values used as camera information.

Table C-5: Typical F-Numbers and corresponding Aperture values

F-Number	1	1.4	2	2.8	4	5.6	8	11	16	22	32
Aperture Value	0	1	2	3	4	5	6	7	8	9	10

Note that F-Numbers and Aperture Values have the following relation.

Aperture Value (Av) = 2 log2 (F-Number) (2)

See also: Aperture Value

Aperture Value: This field specifies the lens aperture (Av) used when the image was captured. The value of this field is `Real` in APEX units.

See also: F-Number

Exposure Program: This field specifies the class of exposure program that the camera used at the time the image was captured. The value of this field is encoded as a `String`. Note the definitions for the following program modes, however, values are camera and vendor specific:

- `Manual` means the exposure setting is set manually by the photographer
- `Program Normal` is a general purpose auto-exposure
- `Aperture Priority` means that the user selected the aperture and the camera selected the shutter speed for proper exposure
- `Shutter Priority` means that the user selected the shutter speed and the camera selected the aperture for proper exposure
- `Program Creative` is biased toward greater depth of field
- `Program Action` is biased toward faster shutter speed
- `Portrait Mode` is intended for close-up photos with the background out of focus
- `Landscape Mode` is intended for landscapes with the background in good focus

Brightness Value: This field specifies the Brightness Value (Bv) measured when the image was captured, using APEX units. The expected maximum value is approximately 13.00 corresponding to a picture taken of a snow scene on a sunny day, and the expected minimum value is approximately –3.00 corresponding to a night scene. If the value supplied by the capture device represents a range of values rather than a single value, the minimum and maximum value may be specified. These values must be `Real`.

Exposure Bias Value: This field specifies the actual exposure bias (the amount of over or under-exposure relative to a normal exposure, as determined by the camera's exposure system) used when capturing the image, using APEX units. The value is the number of exposure values (stops). For example, -1.00 indicates 1 eV (1 stop) underexposure, or half the normal exposure. The value of this field must be a `Real`.

Subject Distance: This field specifies the distance between the front nodal plane of the lens and the subject on which the camera was focusing. Note that the camera may have focused on a subject within the scene that may not have been the primary subject. The subject distance may be specified by a single number if the exact value is known. Alternatively, a range of values indicating the minimum and maximum distance of the subject may be set. The value of this field must be a `Real` in meters.

Metering Mode: This field specifies the metering mode (the camera's method of spatially weighting the scene luminance values to determine the sensor exposure) used when capturing the image. The value of this field is encoded as a `String`. Possible values include "Average", "Center Weighted Average", "Spot", "MultiSpot", "Pattern" and "Partial", however, values are camera and vendor specific.

Scene Illuminant: This field specifies the light source (scene illuminant) that was present when the image was captured. The value of this field is encoded as a `String`. Valid values but not limited to include; "Daylight", "Fluorescent light", "Tungsten Lamp", "Flash", "Standard Illuminant A", "Standard Illuminant B", "Standard Illuminant C", "D55 Illuminant", "D65 Illuminant", "D75 Illuminant", however, values are camera and vendor specific.

Color Temperature: This field specifies the actual color temperature value of the scene illuminant stored in units of Kelvin. The value of this field must be a `Non-negative Real`.

Focal Length: This field specifies the lens focal length used to capture the image. The focal length may be specified by using a single number, for a fixed focal length lens or a zoom lens, if the zoom position is know. Alternatively, two values, the minimum and maximum focal length, may be used to indicate the range of uncertainty in the focal length setting. The value of this field must be a `Positive Real` in meters.

Flash: This field specifies whether flash was used at image capture. The value of this field is `Boolean`.

Flash Energy: This field specifies the amount of flash energy that was used. The measurement units are Beam Candle Power Seconds (BCPS). The value of this field must be a `Non-negative Real`.

Flash Return: This field specifies whether the camera judged that the flash was not effective at the time of exposure. The value of this field is `Boolean`.

Back Light: This field specifies the camera's evaluation of the lighting conditions at the time of exposure. The value of this field is encoded as a `String`. Note the following definitions for lighting situations:

- `Front Light` means the subject is illuminated from the front side.
- `Back Light 1` means the brightness value difference between the subject center and the surrounding area is greater than one full step (APEX). The frame is exposed for the subject center.
- `Back Light 2` means the brightness value difference between the subject center and the surrounding area is greater than one full step (APEX). The frame is exposed for the surrounding area.

Subject Position: This field specifies the approximate location of the subject in the scene. The subject location may be specified as a Position defined in Position.

Exposure Index: This field specifies the exposure index setting the camera selected. The value of this field must be a `Real`.

Auto Focus: This field specifies the status of the focus of the capture device at the time of capture. The value of this field is encoded as a `String`. Note the following definitions for auto focus, however, values are camera and vendor specific:

- `Auto Focus Used` means that the camera successfully focused on the subject.
- `Auto Focus Interrupted` means that the image was captured before the camera had successfully focused on the subject.
- `Near Focused` means that the camera deliberately focused at a distance closer than the subject to allow for the super-imposition of a focused foreground subject.
- `Soft Focused` means that the camera deliberately did not focus exactly at the subject distance to create a softer image (commonly used for portraits).

- `Manual` means that the camera was focused manually

Special Effects: This field specifies the types of special effects filters used. It contains a list of filter fields, where the order of the fields in the array indicates the stacking order of the filters. The first value in the array is the filter closest to the original scene. The value of this field is encoded as a `String`. Possible values; "None", "Colored", "Diffusion", "Multi-image", "Polarizing", "Split-field", "Star", however, values are camera and vendor specific.

Camera Location: This field specifies the location of the camera when the picture was taken. See Location for the format of this field.

Orientation: This field specifies the orientation of the camera when the picture was taken. See Direction for the format of this field.

Print Aspect Ratio (PAR): This field specifies the print aspect ratio specified by the user when the picture was taken. See Rectangular Region for the format of this field.

C.3.2.6 Accessories

Professional and amateur photographers may want to keep track of a variety of miscellaneous technical information, such as the use of extension tubes, bellows, close-up lenses, and other specialized accessories.

See Product Description for the format of this field.

C.3.3 Scanner Capture Metadata

This section specifies scanner capture metadata that may be used for various scanners such as flatbed and film scanners. It optionally contains scanner information, device characterization and scanner capture settings.

C.3.3.1 Scanner Information

This section contains information about a particular scanner that was used to digitize an image item. It is recommended that applications are able to create a unique value of the scanner by combining all fields.

See Product Description for the format of this field.

C.3.3.2 Software Information

This section specifies information about the software or firmware used to capture the image.

See Product Description for the format of this field.

C.3.3.3 Scanner Capture Settings

This section describes the scanner settings used when the image was scanned.

Pixel Size: This field specifies the pixel size, in micrometers, of the scanner. The value of this field must be encoded as a `Real`.

Physical Scan Resolution: These field specify the physical scanning resolution of the device (not the interpolated resolution of the final output data) in the X and Y directions. The value of these fields must be encoded as `Reals` and units are in meters.

C.3.4 Captured Item Metadata

This section specifies capture item metadata. It optionally contains reflection print or film.

C.3.4.1 Reflection Print

This section contains information about a reflection print that was digitally captured. It optionally contains a `Printed Item`, which describes what was used to create this reflection print.

Document Size: This filed specifies the lengths of the X and Y dimension of the original photograph or document, respectively. The values of these fields must be encoded as `Reals`, and are given in meters.

Medium: This filed specifies the medium of the original photograph, document, or artifact. This field must be encoded as a `String`.

Type: This filed specifies the type of the original document or photographic print. This field must be encoded as a `String`.

C.3.4.2 Film

This section contains a description of a piece of film that was digitized.

Brand: This field specifies the name of the film manufacturer, the brand name, product code and generation code (for example, Acme Bronze 100, Acme Aerial 100). This field must be encoded as a `String`.

Category: This field specifies the category of film used. Note: The category Chromagenic refers to B/W negative film that is developed with a C41 process (i.e., color negative chemistry). This field must be encoded as a `String`.

Film Size: This field specifies the size of the X and Y dimension of the film used, and the unit is in meters. The values of these fields must be encoded as `Real`s.

Roll ID: This filed specifies the roll number or ID of the film. For some film, this number is encoded on the film cartridge as a bar code. This field must be encoded as a `Positive Integer`.

Frame ID: This filed specifies the frame number or ID of the frame digitized from the roll of film. This field must be encoded as a `Positive Integer`.

Film Speed: This field specifies the film speed of the film. This filed must be encoded as a `Positive Integer`.

Annex D: Content Description Metadata

D.1 Overview

This annex comprises the content description of an image. The content description has two main purposes:

Firstly – it can be used to classify the image. Images placed in a database need to be extracted from that database. For any application (happy snaps saved in the file system of a personal computer through to professional photo library), this is required. This classification can be used to search for images.

Secondly – once an image is retrieved, some data (that is not useful when searching) may be included with the image that describes the image. For example – "Craig is the guy asleep on the lounge" is not all that useful when searching, but is useful when describing the content.

The metadata listed in this annex contains data for both of the above cases.

All fields listed in this annex are optional unless otherwise stated. A DIG35 compliant metadata reader / editor must understand all fields unless otherwise stated. Note that a DIG35 metadata editor should not remove fields that are not understood when a DIG metadata file is modified.

D.2 Structure

The following diagram illustrates the logical structure of the Content Description metadata.

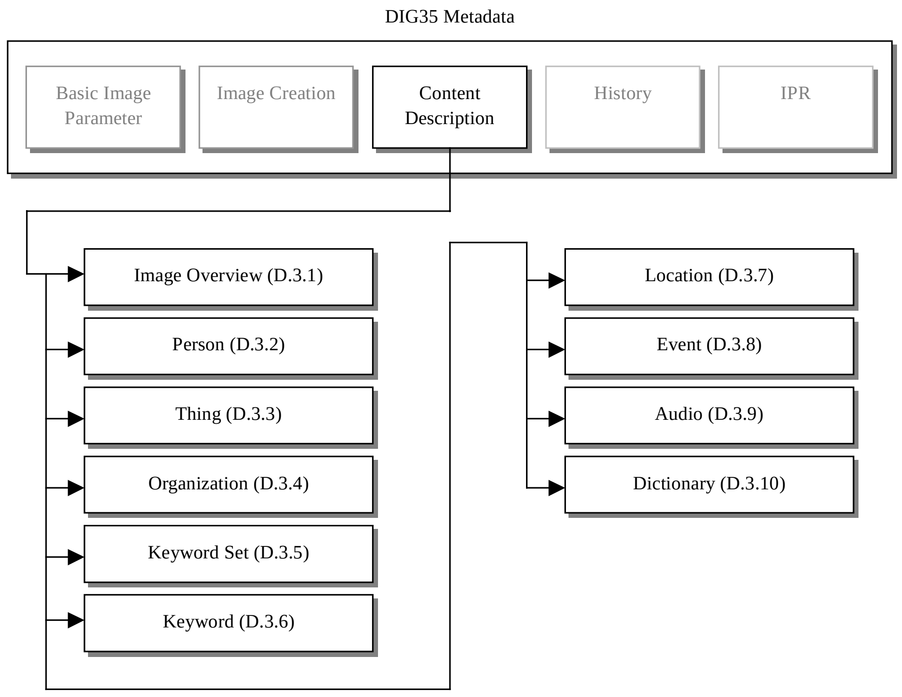


Figure D-1: Content Description metadata structure

D.3 Definition

D.3.1 Image Overview

This section defines general metadata that describes the image. Each field may contain a Timestamp and a Language Identifier.

D.3.1.1 Roll Caption

This field contains text that describes the subject or purpose of a group or roll of images (e.g., a roll of film). The image in the digital file is one member of the "roll."

D.3.1.2 Caption

This field contains text that describes the subject or purpose of the image. It may be additionally used to provide any other type of information related to the image.

D.3.1.3 Capture Time and Date

This field specifies the time & date the image was initially generated. This may be different to the capture device date where the capture device is a scanner that scans the image at a different time to when it was initially captured. The time & date field may contain a cross-reference to the capture device date.

See DateTime for the format of this field.

D.3.1.4 Comment

This field contains additional user/application defined information beyond the scope of other properties in this sub-block.

D.3.2 Person Description

This section lists the fields that can be used to describe a person within an image. This description can also be used to describe the entire image. The content description can contain a Person Description for the entire image and / or a single or multiple descriptions for objects within the image. The person description contains the description of a person in the image.

See Person Description for the format of this field.

D.3.2.1 Position

The position of the person within the image can be specified. The format of this field is defined in Position.

D.3.2.2 Location

The physical location of a person within the image can be specified. Note that this field does not specify the relative position of the person within the image. The format of this field is defined in Location.

D.3.2.3 Keyword / Keyword Set

The Person Description can also contain a keyword or keyword set (Keyword, Keyword Set).

D.3.3 Thing

This field contains text that specifies the names of tangible objects depicted in the image (Washington Monument, for example). Multiple entries are allowed. This field can contain Location and Position field. This field may have a Timestamp and a Language Identifier.

D.3.3.1 Subthings

The Thing field may contain zero or more Thing fields, with the interpretation that these are sub-things of the containing Thing.

D.3.4 Organization Description

This section lists the fields that can be used to describe an organization within an image. This description can also be used to describe the entire image. The content description can contain an Organization Description for the entire image and / or a single or multiple descriptions for objects within the image. The organization description contains the description of an organization in the image.

See Organization Description for the format of this field.

D.3.4.1 Position

The position of the organization within the image can be specified. The format of this field is defined in Position.

D.3.4.2 Location

The physical location of a organization within the image can be specified. Note that this field does not specify the relative position of the organization within the image. The format of this field is defined in Location.

D.3.4.3 Keyword / Keyword Set

The Organization Description can also contain a keyword or keyword set (Keyword, Keyword Set).

D.3.5 Keyword Set

A Keyword Set contains one or more keyword sets and keywords. Keyword sets allow hierarchy of keywords.

An example of a keyword set / keyword hierarchy:

Items in the sky:	⇐ **Keyword Set**
Birds, Planes, Superman	⇐ **Keywords**
Items on the beach:	⇐ **Keyword Set**
Under the umbrella:	⇐ **Keyword Set**
Beach Towel, Spade, Can of Coke	⇐ **Keywords**
Lifeguard	⇐ **Keywords**
Windsurfer	⇐ **Keywords**
Tree, Wattle, Beach Scene	⇐ **Keywords**

The Keyword set can contain a Timestamp and a Language Identifier.

The following fields can also be contained in a keyword set:

D.3.5.1 Title

This field is a `String` . It contains a list of keyword describing either the image or part thereof.

D.3.5.2 Dictionary Reference

The keyword set can also contain a reference to a dictionary (see D.3.10 Dictionary Definition).

D.3.5.3 Comment

A text field for describing the keyword set.

D.3.5.4 Keyword / Keyword Set

The keyword set can also contain a keyword or keyword set (Keyword, Keyword Set).

D.3.6 Keyword

A keyword is either a single word or a small phrase. The keyword is a non-exact language-specific definition of the image or part of the image. The Keyword can contain a Timestamp and a Language Identifier. A keyword field can contain the following fields:

D.3.6.1 Title

A text field which is the keyword itself.

D.3.6.2 Dictionary Reference

The keyword set can also contain a reference to a dictionary (see D.3.10 Dictionary Definition).

D.3.6.3 Comment

A text field for describing the keyword.

D.3.7 Location

The section describes the location of the image. This location is the physical location of the object (e.g. address, GPS coordinate), not the position of an object within the image.

See Location for the format of this field.

D.3.8 Event

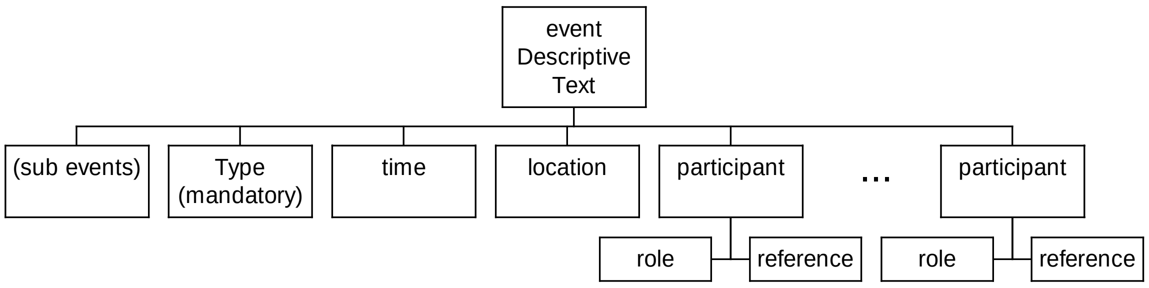
This field contains a description of the events depicted in the image. Events may be personal or societal (e.g., birthday, anniversary, New Year's Eve). Editorial applications may use this property to describe historical, political, or natural events (e.g., a coronation, the Crimean War, Hurricane Andrew).

The event is described by a tree of event fields. The root might represent either an instantaneous event (e.g. Anne dances in a competition) with no sub-events, or else an event with significant duration (e.g., a family reunion). The level below the root might represent the various phases in an event with duration (e.g. arrivals, games, the picnic dinner, after-dinner conversations, and departures for the family reunion). A leaf level event might represent a specific instant (e.g. a particularly impressive spike in the volleyball game before the dinner at the reunion).

Each event field has sub-fields to define or describe:

- the event type or action;
- the time of the event;
- the location of the event;
- the participants in the event;
- the roles of the participants.

The following diagram shows the structure of the event field.



This field may contain a Timestamp and a Language Identifier.

D.3.8.1 Event Type

The Event field contains exactly one Event Type field. This is a mandatory field.

The field is a `String` that specifies the event.

Examples:

- Wedding
- Birthday

- Holiday

This field may contain a Language Identifier.

D.3.8.2 Participant

The event field contains zero or more Participant fields.

Role: The Participant field has exactly one Role field. This field is mandatory.

The Role field is a `String`. The role is the role that the participant has in the event. Examples:

- Chair
- President
- Goal Keeper

Object Reference: The Participant field has exactly one Object Reference field. This field is mandatory.

The Object Reference field contains text that refers to a valid identifier field of one of the People, Organization or Things fields.

D.3.8.3 Time & Date

This is the time & date of the event. This field is a DateTime.

D.3.8.4 Duration

This is the duration of the event.

D.3.8.5 Location

The location of the event. This location is the physical location of the event and not the position within the image. This field is a Location.

D.3.8.6 Comment

This is a `String` description of the event. This field is used to describe an event where the other event fields listed are not suitable.

D.3.8.7 Subevents

The Event field may contain zero or more Event fields, with the interpretation that these are sub-events of the containing Event.

D.3.9 Audio

Image metadata can contain one or more audio streams. Each audio stream can contain a comment field describing the audio. A single comment should also be able to describe more than one audio stream. The Audio field can contain a Timestamp and a Language Identifier.

D.3.9.1 Audio Stream

This field contains an audio stream. The format of the stream is not defined.

D.3.9.2 Comment

This field is used to describe the audio annotation. The field is a `String`.

D.3.10 Dictionary Definition

There is no known list of dictionaries. A dictionary definition is just stored as a `String` which is the name of the dictionary used to define a keyword.

Annex E: History Metadata

E.1 Overview

This annex comprises the history of an image. The History metadata is used to provide *partial information about* how the picture got to the present state. This data is only approximate because;

- some of the data is collapsed, thus providing only a summary
- some of the data may not have been properly entered because applications weren't written to update the history metadata.

The history metadata contains;

- a summary of image editing operations that have already been applied to the picture;
- the history of the image creation metadata.

E.2 Structure

The following diagram illustrates the logical structure of the History metadata.

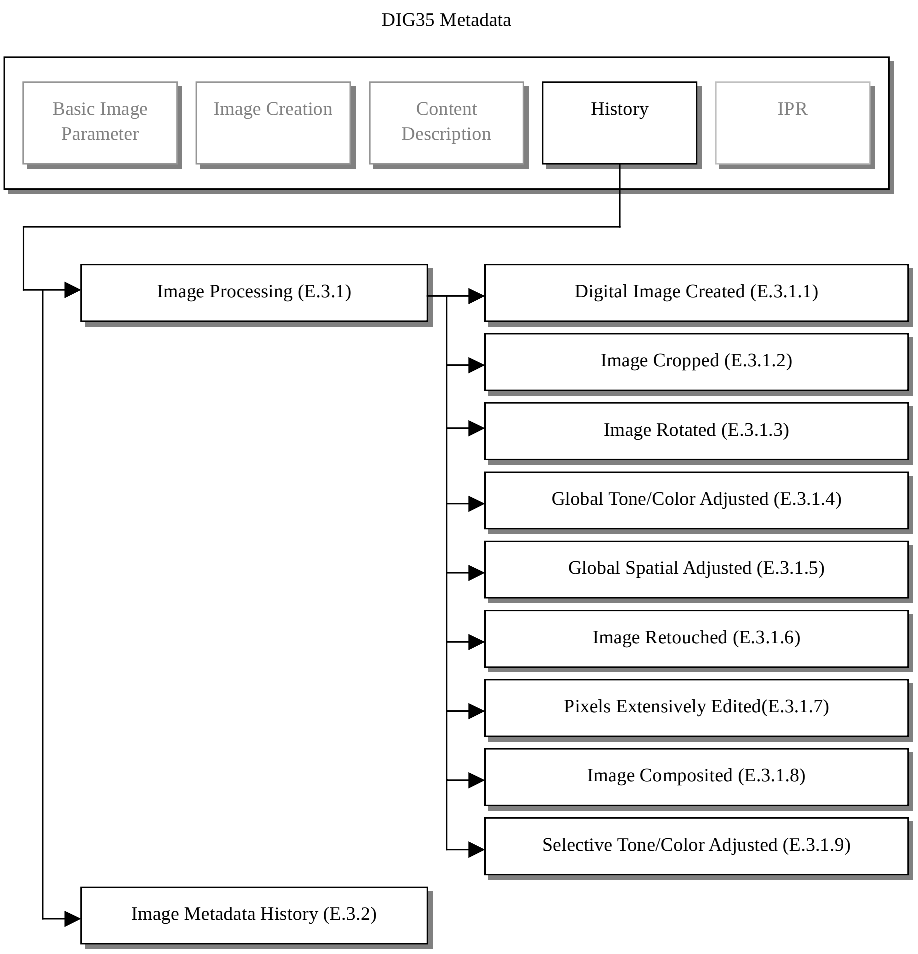

DIG35 Metadata
Basic Image Parameter
Image Creation
Content Description
History
IPR
Image Processing (E.3.1)
Digital Image Created (E.3.1.1)
Image Cropped (E.3.1.2)
Image Rotated (E.3.1.3)
Global Tone/Color Adjusted (E.3.1.4)
Global Spatial Adjusted (E.3.1.5)
Image Retouched (E.3.1.6)
Pixels Extensively Edited(E.3.1.7)
Image Composited (E.3.1.8)
Selective Tone/Color Adjusted (E.3.1.9)
Image Metadata History (E.3.2)

Figure E-1: History metadata structure

E.2.1 Reasons For Including History Metadata

For active image editors, the history/digital processing metadata can help identify likely images that correspond to original images (the digital negatives). Or conversely, the metadata may help to identify the likely “final result” images. For cooperative and collaborative image editing, or just for convenience in work flow management, this metadata can identify the “next processing step” to be applied. This data can improve efficiency by indicating steps that have been applied and allowing subsequent processing to avoid repetitive or unnecessary editing steps. Finally, this processing may help to avoid processing steps that could degrade the image quality. If a printing system provides automatic sharpening (or blurring) capability, it is sometimes prudent to turn off the automatic sharpening if the image has been blurred or sharpened already, or if the pixels have been re-sampled to convert to a different resolution.

The image creation history records all but the last image creation metadata. In some cases, an image may have a series of creation events including digital capture, exposure of negative or reversal films, creation of prints, transmissive scans of negatives or positive film, or reflective scans of prints. All of this metadata is important for some applications.

The history metadata is a container for the Digital Processing Metadata and the Image Creation Metadata history. The history metadata may contain a Timestamp field indicating when it was last changed.

E.3 Definition

E.3.1 Image Processing Hints Metadata

The image processing hints defined below should be considered single-valued flags. That is because the presence of a particular hint, such as “Image Cropped” indicates that the image has been cropped. Absence of a “Image Cropped” hint is no assurance that the image has never been cropped, however.

This point deserves some further elaboration. ***The flags provide no indication of the details of the processing.***[i] For instance, suppose that a reflective print was scanned and the resulting image edited. If the cropping during the pre-scan was not perfect, the user might need to remove – by cropping – the small white border. The “Image Cropped” flag would be set, but in this case (likely not too common) the *de minimus* cropping would be reported just the same as cropping that removed whole objects from the digital image.

With the philosophy explained above, there is an important consequence: ***a particular image editing hint will appear at most once.*** If there is no editing script or description, there is no use in knowing that a image was cropped twice. A twice-cropped picture is in every way equivalent to an image cropped only once, but to the final size.

The following subsections define each item in the editing hints metadata.

E.3.1.1 Digital Image Created

This flag indicates that a image was created by a metadata-aware application or process. Clearly a image with this flag would have been created, so the information is in the “by a metadata-aware application or process”. In encodings for certain file formats, this flag provides a place-holder that makes it easy for applications to add other editing hint flags. Digital processing steps that are part and parcel of the image creation process **must not be flagged.** For instance, a process to create images by scanning negatives may crop the digital picture to remove borders, may adjust brightness and contrast, may rotate the image under operator control, may reduce red-eye effects, and may compress the image to create an Exif file. All of these are part of the creation process, and are not separately flagged.

E.3.1.2 Image Cropped

The presence of this flag indicates that a image editing application, program, or system has cropped the image. For the purpose of this flag, any cropping is significant and should be recorded.

E.3.1.3 Image Rotated

The presence of this flag indicates that a image has been rotated. While some rotations are reversible, but others are irreversible, both kinds are treated identically, causing the presence of the Image Rotated flag. This flag is also set after a flip or mirror operation.

[i] At least, at the present time they don’t. The last section, on extensions, indicates some avenues for extending this metadata.

E.3.1.4 Global Tone/Color Adjustment Applied

The presence of this flag indicates that a contrast or density adjustment has been applied to the image, or that the image coloring has been adjusted.

E.3.1.5 Global Spatial Adjustment Applied

The presence of this flag indicates that the image has been sharpened, or compressed, or blurred, or re-sampled.

E.3.1.6 Image Retouched

The presence of this flag indicates the image pixels have been edited to remove scratches or redeye, or other minor image blemishes.

E.3.1.7 Pixels Extensively Edited

The presence of this flag indicates the image has been edited extensively – enough to change the captured scene content.

E.3.1.8 Image Composited With Another Image, Background, Graphics, or Text

The presence of this flag indicates the image has been created by compositing a picture with another picture, or a background, graphic, or text.

E.3.1.9 Selective Tone/Color Adjustment Applied

The presence of this flag indicates that a contrast or density adjustment has been applied to a selected region of the image.

E.3.2 Image Metadata History

Each time a new image is created, some of the metadata from the previous image may be moved to, or else referenced by, the image metadata history. The previous image's Basic Image Parameter, Image Creation, Content Description, and IPR metadata can be recorded in a Metadata History node. Additionally, when image persistence is assured, a Universal Image ID (UIID) may be recorded, identifying the previous image. For instance, a slide exposure, followed by creation of an internegative, followed by a scan of the internegative could result in an image with the metadata relevant to the slide and internegative recorded in repeated Metadata History nodes. The internegative scan metadata would be in the Image Creation metadata.

When a new image is created by compositing several images, the Image Metadata History has a hierarchical structure. There is a top level Metadata History node, occupying the usual position in the list of repeated Metadata History nodes. That top level Metadata History node contains only other Metadata History nodes, however. In fact, it must contain at least two Metadata History nodes, reflecting that fact that single image compositing operations make no sense. Each of the contained nodes may itself contain a UIID, or one or more nodes with Basic Image Parameters, Image Creation, Content Description, or IPR metadata. A contained node may alternatively contain another set of at least two Metadata History Nodes, representing successive compositing operations.

The recursive/repeated structure of the Metadata History nodes in the History metadata can be understood with the following rule: serial image processing operations result in repeated Metadata History nodes. Image Compositing results in a Metadata History node that contains the source image metadata history.

Each Metadata History node has exactly one of the following structural alternatives:

- It may contain a single UIID
- It may contain at most one Basic Image Parameter metadata structure and at most one Image Creation metadata structure and at most one Content Description metadata structure and at most one IPR metadata structure. In this case all of the metadata structures must be from the same image, which is a previous version of the current image.
- It may contain two or more Metadata History nodes.

Each of these alternatives is represented in the Figure E-2.

When a new image is created because a previous image was edited, a new Metadata History node is appended to the list of Metadata History nodes (or if there are no Metadata History nodes, the first is created). The newly-created Metadata History node is populated by copying in a UIID representing the source image, or else by copying in one or more of the Basic Image Parameter, Image Creation, Content Description, or IPR metadata structures from the previous image.

When a new image is created by compositing several previous images, the new image History metadata will have a single Metadata History node. That (top level) node will contain at least two Metadata History nodes, which are created from the two or more previous images that are being composited.

Note: because of the structure of DTD element definitions, the above text takes precedence over the limited syntax rules in the DTD.

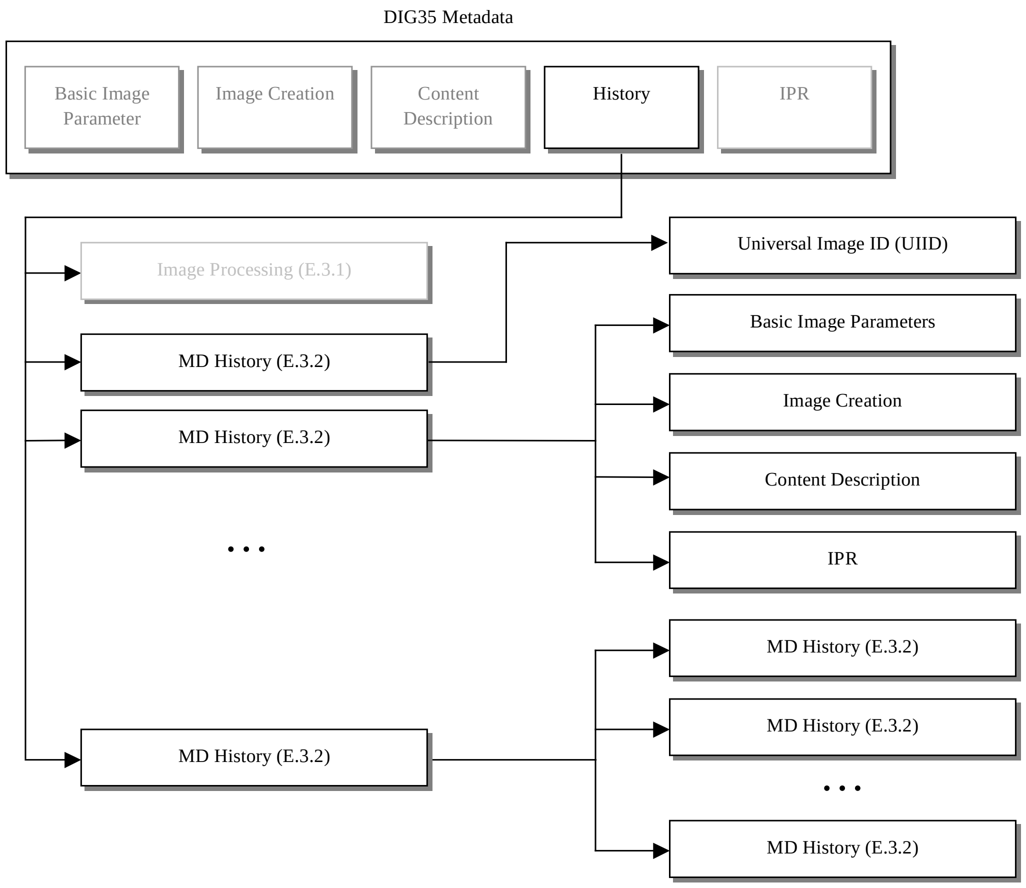


Figure E-2: Image Creation History metadata structure

Annex F: IPR Metadata

F.1 Overview

Intellectual Property Rights (IPR) metadata must preserve both moral rights and copyrights. In addition, more information may be required to be added, such as conditions of use (i.e., mainly limitations attached to the purchasing or licensing of copyrights). Names, content description, dates, etc. may be useful to establish the priority of IPR if different vendors propose an image.

To ease the processing of IPR-related administrative tasks, identification (e.g., a unique inventory number) and contact point for exploitation are also considered useful metadata.

F.2 Structure

The following diagram illustrates the logical structure of the IPR metadata.

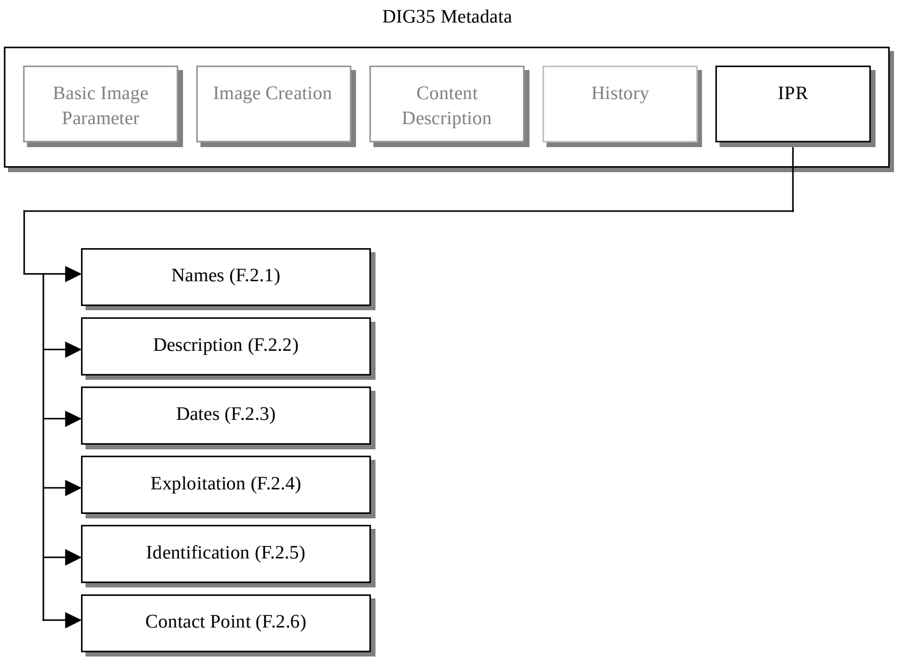


Figure F-1: Content description metadata structure

F.3 Definition

F.3.1 Names

This section specifies names related to the original work.

Persons appearing on the image should be named, because there are restrictions on publishing the image of a person who does not agree with publication. Who what and where (i.e., the subject of the image) can also be the title of the image. All fields are a `String`. Each field may contain a Timestamp and a Language Identifier.

F.3.1.1 Original Work Author

This field specifies the name of the author who created the original work (e.g., painter sculptor, architect, etc.).

F.3.1.2 Image Creator

This field specifies the name of the image creator. The image creator could be, for example, the photographer who captured the original picture on film, the illustrator, or graphic artist who conducted the image-creation process, etc.

F.3.1.3 Right Holder

This field specifies the name of the intellectual property right holder of the image. The right holder may be the author of the image, a stock photo agency, or vendor.

F.3.2 Description

This field specifies the description of the content. The format is vendor specific. All fields are a `String`. Each field may contain a Timestamp and a Language Identifier.

F.3.2.1 Title

This field specifies the title of the image.

The title is given by the author, thereby adding meaning to the image in many occasions. However some titles are not significant of IPR, e.g., "portrait of a man.” It may be considered that this type of title has been added by the curator of the work and not by the author himself.

F.3.2.2 Legend

This field specifies the legend.

This is a more detailed description of what appears on the image. This may also be the caption written by a photographer on the back of a photographic print. This field may answer the question, "why?"

F.3.2.3 Caption

This field specifies the caption of the image.

This field addresses the text which has been added as complementary information to assist in understanding the image’s content (e.g., second draft by Durer for a study on a Biblical scene). The caption often has a tutorial motivation.

F.3.2.4 Copyright

This field specifies the copyright notice of the image.

The copyright notice is mandatory to appear on the side of a photograph when printed or published.

F.3.3 Dates

This section specifies the IPR-related date information. All values must be a valid DateTime type.

There are a variety of valid DateTime formats. For example, a date may be an exact year, possibly with month and day, sometimes with hour, minute, second and thousandth (i.e., ISO timestamp, which is always GMT time). However, date may also be less delimiting. For example, the date could be "first half of the fifteenth century," "late middle-age," "early Roman," etc

Professional applications may prefer an exact date, whereas specifying a year +/- 5 years may satisfy users of early century photographs.

F.3.3.1 Original Work Creation Date

This field specifies the date that the original work was created.

F.3.3.2 Picture Taken Date

This field specifies the date that the picture was taken.

F.3.3.3 Scan Date

This field specifies the date that the image was scanned.

F.3.3.4 Processing Date

This field specifies the date that the image was processed.

F.3.3.5 Modification Date

This field specifies the date when any kind of modification was made to the original work. This field is to be concurrently stored as an operation in the history log.

F.3.3.6 Last Modification Date

This field specifies the last date the image was modified.

F.3.4 Exploitation

Registration, resulting in delivery of a License Plate or unique identifier, is one of the most important phases in protection of the content. Watermarking is now required by the intellectual property right holders in electronic commerce applications. In fact, it is a dissuasive protection for which owners are looking, because the watermark is invisible to the viewer and therefore may not be dissuasive to the pirate.

F.3.4.1 Protection

Protection either indicates that there is a watermark, that the image is registered or protected by any means.

This field is an `Integer`. The field may contain a Timestamp.

A value of zero specifies that the image contains no watermark. Values between 1 and 255 are reserved for JURA use. If this fields is not present, the watermark content (or it's presence) is undefined.

F.3.4.2 Restriction of Use

Restrictions of use may apply to an image that is not allowed outside the factory for industrial applications, or for which exclusive rights of copy have been delegated to a unique agency, or for which prior authorization of represented people is mandatory before publishing.

This field is a `String`. The field may contain a Timestamp and a Language Identifier.

F.3.4.3 Obligations

Obligation may concern the copyright mention addressed elsewhere.

This field is a `String`. The field may contain a Timestamp and a Language Identifier.

F.3.4.4 IPR Management System

IPR Management Systems such as IPMP (Intellectual Property Management & Protection) or ECMS (Electronic Copyright Management System) use these fields to determine where information is kept regarding the management system.

An example use of these fields is to track the usage of an image. During transfer, an agency determines the owner of the image from the management systems fields. It already knows the consumer, and uses this information to charge the user and credit the owner the amount as determined by the management system.

These systems are used to track images. Information is stored on a server describing the IPR of the image, and depending is mandatory or recommended, there must be a link to where all information about it is kept.

The field may contain a Timestamp and a Language Identifier.

Type: The IPR Management System being used. This field is a `String`.

ID: Information of an ID. This field is a `String`.

Location: Information of the location. This field is a `Reference`.

F.3.5 Identification

This is a link to a place (e.g., secured database or other storage place) where critical information is kept. The identifier "identifies" a content, which means only one; therefore, if an image is cropped, modified or made a new image, then to the image must be registered again, and a new identifier must be acquired, because there are now two objects instead of one. However, the parent image must appear in the metadata set of the child.

F.3.5.1 Generic IPR Identifier

Identification mode describes which type of identification is applied to the content.

The field may contain a Timestamp and a Language Identifier.

Mode: This field describes the identification mode. This field is a `String`.

ID: This is the identification. The field content is described by the mode field. This field is a `String`.

F.3.5.2 License Plate

These fields specify the license plate of the original image, defined in ISO 10918-3. The combination of the fields in the license plate contain a globally unique number.

These fields may contain a Timestamp and a Language Identifier.

Country: This field is an `Identifier`. It contains the country of the address. The field contains the country code for the license plate as defined in [ISO 3166-1].

Registration Authority: This field is an `Integer`. The field contains the registration authority code for the license plate.

Registration Number: This field is an `Integer`. The field contains the registration code for the license plate.

F.3.6 Contact Point

This section specifies the contact point of the right holder.

F.3.6.1 Address

This specifies the postal, phone and fax contacts for a potential buyer to get all required information and possibly the order form.

F.3.6.2 Link

This specifies either an email address or a URL.

F.3.6.3 Collection

This field is a link to a collector, museum, group, institution, etc. URL.

F.4 Discussion Items

Intellectual property rights are of significant concern, because there are serious ramifications to an end user's actions regarding images. Publication, alteration, modification, etc. can have profound legal consequences. Also, there may be a series of rights that originate from a single, original image. Accordingly, the following is a non-exhaustive list of discussion items related to IPR:

- How much of the IRP metadata actually interests the personal applications, the professional applications?
- What is the IPR importance of the operator for processing, editing, collaging etc. What is the IPR importance of the operator for scanning, color-correcting, as well as all types of enhancements?

Annex G: XML Encoding Rules and Guidelines

This annex specifies the DIG35 XML document structure as well as the fundamental metadata types and field defined in Annex A.

G.1 DIG35 Document Type Definition Overview

G.1.1 DIG35 metadata document

An XML document that contains a "METADATA" element defined in this specification is called a *DIG35 metadata document*. A *DIG35 metadata document* may either be an independent XML document, that may contain other XML fragments with a different XML namespace, or be a fragment itself being a sub-tree of a parent XML document. The former type is called a *stand-alone DIG35 metadata document* where the root element starts with the "METADATA" element and the latter type is called a *fragmented DIG35 metadata document*. In both cases, it is expected that DIG35 metadata elements and other XML element definitions can be distinguished by each XML namespace.

G.1.2 Structure

A DIG35 metadata document is designed to describe metadata for either a single image or a collection of images. Single image metadata would contain, for example, the camera capture information (e.g. date, time, shutter speed, etc.) or a description of the objects (e.g. people and things) for that image. On the other hand, metadata for a collection of images would be information that ties the images together, as a group, such that the collection itself comprises a semantic meaning. Such metadata would be, for example, a group of pictures taken at an event (e.g. a wedding or a birthday party) or the selection criteria of an image database search. Note that each image within a collection may also contain individual metadata and can be recorded in the same metadata document as the image collection metadata. The individual metadata would be stored as a sub-tree of the image collection metadata document. This means that the "METADATA" element may also be nested. Metadata for collections may contain multiple metadata relative to a single image or even other metadata for a different collection of images constructing a tree structure. However, metadata for a single image may not contain other image metadata therefore considered as a leaf node.

It is important to note that the scope of the metadata applicable is bound by the "METADATA" element.

G.1.3 DIG35 Namespace

XML namespace is a collection of names, identified by a Universal Resource Identifier (URI), that allows XML documents of different sources to use elements with the same names, to be merged within a single document with no confusion. Considering DIG35 metadata either incorporating other metadata for extensibility or being used in other applications, it is important to define a XML namespace for DIG35 elements and attributes.

To specify the DIG35 XML namespace the following URI is defined. It should be used by experimental implementations:

```
xmlns=“http://www.digitalimaging.org/init/dig35/wd1.0”
```

Note: This value will be replaced with a proper URI when this working draft is published as a DIG specification.

G.1.4 The "METADATA" Element

The "METADATA" Element is defined as follows.

The TYPE attribute value "`SINGLE`" or "`COLLECTION`" is used to specify whether the METADATA element is describing a single image or a collection of images, respectively.

```
METADATA              ::= '<' 'METADATA' language? timestamp? TYPE? '>'
                              (
                                    BASIC_PARAM?
                                    CREATION?
                                    CONTENT?
                                    HISTORY?
                                    IPR?
                              )
                              METADATA*
                            '</METADATA>'
TYPE                  ::= 'TYPE' '=' '"SINGLE"' | '"COLLECTION"'
```

G.1.5 DIG35 Metadata Document Examples

The following example shows a 'stand-alone DIG35 metadata document' for a single image.

```
<?xml version="1.0" ?>
<METADATA TYPE="SINGLE" xmlns="http://www.digitalimaging.org/init/dig35/wd1.0">
    <!-- The DIG35 XML document; each DIG35 element with no namespace prefix
    -->
</METADATA>
```

The next example shows metadata for a collection of images with two images having its own metadata as well.

```
<?xml version=`1.0' ?>
<METADATA TYPE="COLLECTION">
    <CREATION>
        <!-- Metadata on how all images were created -->
    </CREATION>
    <CONTENT>
        <!-- Metadata on the content of all the images For example,
             the event name (e.g. Joe and Mary's wedding)
        -->
    </CONTENT>

    <!--  Metadata for individual images   -->
    <!--  Metadata for joe.jpg   -->
    <METADATA TYPE="PICTURE">
        <BASIC_PARAM>
            <FILE xlink:href="joe.jpg" />
        </BASIC_PARAM>
        <!-- Other metadata on joe.jpg. -->
    </METADATA>
    <!--  Metadata for mary.jpg   -->
    <METADATA TYPE="PICTURE">
        <BASIC_PARAM>
            <FILE xlink:href="mary.jpg" />
        </BASIC_PARAM>
        <!-- Other metadata on mary.jpg
             "Mary with college friends"
        -->
    </METADATA>
    <!--  Metadata for other images  -->
</METADATA>
```

A fragmented DIG35 metadata document may be embedded within other XML documents. The following is an example that shows a DIG35 metadata document within a SVG document.

```
<?xml version="1.0" ?>
<svg:svg xmlns:svg="http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
    <svg:metadata>
        <dig35:METADATA
            xmlns:dig35="http://www.digitalimaging.org/init/dig35/wd1.0">
            <!-- The DIG35 XML section, each DIG35 element with namespace
                 prefix -->
        </dig35:METADATA>
    <svg:metadata>
</svg:svg>
```

G.1.6 Extensibility

This specification defines public metadata that is applicable to general images. Thus, its coverage may not be enough for a particular application domain which needs further detail to be stored. DIG35 metadata documents therefore should allow additional elements to be embedded, keeping the integrity of the overall framework, allowing private metadata to be 'plugged-in' that only a particular systems may be able to understand and use such information.

See the PROPRIETARY element definition for details.

Note: It is the intention of DIG35 to define a method to allow such extensions, however, it is yet to be determined if the proposed method is a suitable method at this time.

G.2 Atomic Data Types

This section defines atomic data types that are referenced by the XML syntax definition sections.

Note: It is not the intention of the DIG35 to define yet another data type schema for XML. When W3C publishes the XML Schema specification as a recommendation, this section is expected to become obsolete.

G.2.1 Digit

This abstract type is used by other types.

```
digit    ::=      ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’
```

G.2.2 Integer

This is the atomic data type `Integer`.

```
integer              ::= (‘+’ | ‘-’)? digit+
```

Examples:

```
100000
-20
65
+12345
```

G.2.3 Non-negative Integer

This is the atomic data type `Non-negative Integer`.

```
non-negative-integer  ::=    ‘+’? digit+
```

Examples:

```
100000
65
0
```

G.2.4 Positive Integer

This is the atomic data type `Positive Integer`.

```
positive-integer   ::= ‘+’? digit+
```

This type has the restriction that it must contain a value 1 or greater.

Examples:

```
100000
65
```

G.2.5 Decimal

This abstract type is used by other types.

```
decimal                ::=    (‘+’ | ‘-’)? (digit+ | (digit* ‘.’ digit+))
```

Examples:

```
-1.23
12678967.543233
+100000.00
```

G.2.6 Real

This is the atomic data type `Real`.

```
real                      ::=    (‘+’ | ‘-’)? decimal ((‘E’ | ‘e’) integer)?
```

Examples:

```
-1.23 e 5
12678967.543233
100000.00 e -10
10
0
```

G.2.7 Non-negative Real

This is the atomic data type `Non-negative Real`.

```
non-negative-real         ::=    (‘+’ | ‘-’)? decimal ((‘E’ | ‘e’) integer)?
```

Examples:

```
+1.23 e 5
12678967.543233
100000.00 e -10
10
0
```

G.2.8 Rational

This is the atomic data type `Rational`.

```
rational                  ::=    (‘+’ | ‘-’)? digit+ (‘/’ digit+)?
```

Examples:

```
-1/23
7/5
+160/10
```

G.2.9 Non-negative Rational

This is the atomic data type `Non-negative Rational`.

```
non-negative-rational   ::=  ‘+’? digit+ (‘/’ digit+)?
```

Examples:

```
1/23
7/5
160/10
```

G.2.10 String

This is the atomic data type `String`.

```
string                    ::=    <character>+
```

Stings contain characters. The XML encoding (and various other XML rules) define the exact charachers that are allowed, and the mapping between character codes and printable characters. For example, if an ASCII encoding is used, then ASCII characters are used. XML allows other encodings including UNICODE.

Examples:

```
Under the table
```

G.2.11 Boolean

This is the atomic data type `Boolean`.

```
boolean                    ::=    ‘true’ | ‘false’
```

Examples:

```
true
false
```

G.3 Value Types

G.3.1 URI

Uniform Resource Identifier (URI) provides a simple and extensible means for identifying a resource on the Web.

```
uri                    ::= string
```

The syntax is defined as a `string` in this specification, however, it must conform to the syntax defined in [RFC2396].

G.3.2 Identifier

This is a value type described in Identifier.

```
identifier             ::= string | uri
```

G.4 Defined Data Types

G.4.1 Date and Time

The elements (and sub-elements) are used to specify metadata described in Date and Time.

```
date-time(N)         ::= ‘<’ N language? timestamp? ‘>’
                            YEAR?
                            MONTH?
                            DAY?
                            SEASON?
                            HOUR?
                            MINUTE?
                            SECOND?
                            COMMENT?
                          ‘</’ N ‘>’
```

```
date(N)              ::= ‘<’ N language? timestamp? ‘>’
                            YEAR?
                            MONTH?
                            DAY?
                            SEASON?
                            PROPRIETARY?
                            COMMENT?
                          ‘</’ N ‘>’
```

```
time(N)              ::= ‘<’ N language? timestamp? ‘>’
                            HOUR?
                            MINUTE?
                            SECOND?
                            PROPRIETARY?
                            COMMENT?
                          ‘</’ N ‘>’
```

```
YEAR                 ::= ‘<YEAR>’
                            integer
                          ‘</YEAR>’
```

```
MONTH                ::= ‘<MONTH>’
                            positive-integer
                          ‘</MONTH>’
```

```
DAY                  ::= ‘<DAY>’
                            positive-integer
                          ‘</DAY>’
```

```
SEASON               ::= ‘<SEASON>’
                            string
                          ‘</SEASON>’
```

```
HOUR                 ::= ‘<HOUR>’
                            positive-integer
                          ‘</HOUR>’
```

```
MINUTE               ::= ‘<MINUTE>’
                            positive-integer
                          ‘</MINUTE>’
```

```
SECOND               ::= ‘<SECOND>’
                            positive-integer
                          ‘</SECOND>’
```

date-time example:

```
<YEAR>1999</YEAR>
<MONTH>11</MONTH>
<DAY>18</DAY>
<HOUR>11</HOUR>
<MINUTE>30</MINUTE>
<SECOND>31</SECOND>
```

date example:

```
<YEAR>1999</YEAR>
<MONTH>11</MONTH>
<DAY>18</DAY>
<SEASON>Summer</SEASON>
```

time example:

```
<HOUR>11</HOUR>
<MINUTE>30</MINUTE>
<SECOND>31</SECOND>
```

G.4.2 Address

The type (and sub-elements) are used to specify metadata described in Address.

```
address(N)          ::= '<' N language? timestamp? '>'
                          ADDR_NAME?
                          ADDR*
                          ( POSTCODE | ZIPCODE )?
                          COUNTRY?
                          TYPE?
                        '</' N '>'
```

```
ADDR_NAME           ::= '<' 'ADDR_NAME' language? timestamp? '>'
                          string
                        '</ADDR_NAME>'
```

```
ADDR                ::= '<' 'ADDR' language?  (TYPE = '"' string '"')? '>'
                          string
                        '</ADDR>'
```

```
POSTCODE            ::= '<POSTCODE>'
                          string
                        '</POSTCODE>'
```

```
ZIPCODE             ::= '<ZIPCODE>'
                          string
                        '</ZIPCODE>'
```

```
COUNTRY             ::= '<' 'COUNTRY' language? '>'
                          string
                        '</COUNTRY>'
```

```
TYPE                ::= '<TYPE>'
                          string
                        '</TYPE>'
```

Example:

```
<ADDR_NAME>DIG Headquarters</ADDR_NAME>
<ADDR TYPE = "unit">10</ADDR>
<ADDR TYPE = "street">23 Fleet Street</ADDR>
<ADDR TYPE = "city">Carlingford</ADDR>
<ADDR TYPE = "state">New South Wales</ADDR>
<POSTCODE>2118</POSTCODE>
<COUNTRY>AU</COUNTRY>
<TYPE>Business</TYPE>
```

G.4.3 Phone Number

The type (and sub-elements) are used to specify metadata described in A.2.3.3 Phone Number.

```
phone(N)            ::= ‘<’ N timestamp? ‘>’
                          COUNTRY_CODE
                          AREA
                          LOCAL
                          TYPE?
                        ‘</’ N ‘>’
```

```
COUNTRY_CODE        ::= ‘<COUNTRY_CODE>’
                          positive-integer
                        ‘</COUNTRY_CODE>’
```

```
AREA                ::= ‘<AREA>’
                          positive-integer
                        ‘</AREA>’
```

```
LOCAL               ::= ‘<LOCAL>’
                          positive-integer
                        ‘</LOCAL>’
```

```
TYPE                ::= ‘<TYPE>’
                          string
                        ‘</TYPE>’
```

Example:

```
<COUNTRY_CODE>61</COUNTRY_CODE>
<AREA>2</AREA>
<LOCAL>92122646</LOCAL>
```

G.4.4 Person / Organization

The type (and sub-elements) are used to specify metadata described in Person and Organization.

```
person(N, EX)       ::= ‘<’ N language? timestamp? ‘>’
                          NAME_TITLE?
                          PERSON_NAME*
                          NICK_NAME?
                          JOB_TITLE?
                          COMPANY_NAME?
                          address(‘ADDRESS’)*
                          phone(‘PHONE’)*
                          EMAIL*
                          WEB*
                          date(‘BIRTH_DATE’)?
                          date(‘AGE’)?
                          PROPRIETARY?
                          COMMENT?
                          EX
                        ‘</’ N ‘>’
```

```
organization(N, EX)   ::=    ‘<’ N language? timestamp? ‘>’
                               ORGANIZATION_NAME?
                               address(‘ADDRESS’)*
                               phone(‘PHONE’)*
                               EMAIL*
                               WEB*
                               PROPRIETARY?
                               COMMENT?
                               EX
                             ‘</’ N ‘>’
```

```
ORGANIZATION_NAME ::= ‘<’ ‘ORGANIZATION_NAME’ language? timestamp? ‘>’
```

```
                                string
                              '</ORGANIZATION_NAME>'

NAME_TITLE           ::= '<' 'NAME_TITLE' language? timestamp?>'
                                string
                              '</NAME_TITLE>'

PERSON_NAME          ::= '<' 'PERSON_NAME' language? timestamp?>'
                                NC+
                              '</PERSON_NAME>'

NC                   ::= '<' 'NC' 'TYPE=' NC:TYPE '>'
                                string
                              '</NC>'

NC:TYPE              ::= 'TYPE=' '“'
                                'prefix' | 'given' | 'family' | 'suffix'
                              '”'

NICK_NAME            ::= '<' 'NICK_NAME' language? timestamp? '>'
                                string
                              '</NICK_NAME>'

JOB_TITLE            ::= '<' 'JOB_TITLE' language? timestamp? '>'
                                string
                              '</JOB_TITLE>'

COMPANY_NAME         ::= '<' 'COMPANY_NAME' language? timestamp? '>'
                                string
                              '</COMPANY_NAME>'

EMAIL                ::= '<' 'EMAIL' language? timestamp?
                                                   ('TYPE=' '“' string '”' )?'>'
                                string
                              '</EMAIL>'

WEB                  ::= '<' 'WEB' language? timestamp? uri-loc?
                                          'TYPE=' '“' string '”' '>'
                              '</WEB>'
```

PERSON example:

```
<PERSON>
    <NAME_TITLE>Duke of Somewhereburgh</NAME_TITLE>
    <PERSON_NAME xml:lang=“en”>
        <NC TYPE=“prefix”>Sir</NC>
        <NC>John</NC>
        <NC>R.</NC>
        <NC TYPE=“family”>Smith</NC>
        <NC TYPE=“suffix”>III</NC>
        <NC TYPE=“suffix”>Esq.</NC>
    </PERSON_NAME>
</PERSON>
```

ORGANIZATION example:

```
<ORGANIZATION>
    <ORGANIZATION_NAME>Digital Imaging Group</ORGANIZATION_NAME>
</ORGANIZATION>
```

G.4.5 Location

The type is used to specify metadata described in .

```
location(N)          ::= '<' N language? timestamp? '>'
                           COORD_LOC?
                           UTM_COORD_LOC?
                           address('ADDRESS')?
                           COMMENT?
                           GPS?
                         '</' N '>'
```

Example:

```
<LOCATION>
      <COORD_LOC>…</COORD_LOC>
      <ADDRESS>…</ADDRESS>
      <COMMENT>…</COMMENT>
      <GPS>…</GPS>
</LOCATION>
```

G.4.5.1 Coordinate Location

The element (and sub-elements) are used to specify metadata described in Coordinate Location.

```
COORD_LOC            ::= '<' 'COORD_LOC' timestamp? '>'
                           LONGITUDE
                           LATITUDE
                         '</UTM_COORD_LOC>'
```

```
LONGITUDE            ::= '<LONGITUDE>'
                           non-negative-real
                         '</LONGITUDE>'
```

```
LATITUDE             ::= '<LATITUDE>'
                           non-negative-real
                         '</LATITUDE>'
```

Examples:

```
<COORD_LOC>
    <LONGITUDE>138.700</LONGITUDE>
    <LATITUDE>35.383</LATITUDE>
</COORD_LOC>
```

G.4.5.2 UTM Coordinate Location

The element (and sub-elements) are used to specify metadata described in UTM Coordinate Location.

```
UTM_COORD_LOC        ::= '<' 'UTM_COORD_LOC' timestamp? '>'
                         ALTITUDE?
                         (
                           DATUM?
                           ZONE
                           UTM_LONGITUDE
                           UTM_LATITUDE
                         )?
                         '</UTM_COORD_LOC>'
```

```
ALTITUDE             ::= '<ALTITUDE>'
                           real
                         '</ALTITUDE>'
```

```
DATUM                ::= '<DATUM>'
                           string
                         '</DATUM>'
```

```
ZONE                 ::= '<ZONE>'
```

```
                             positive-integer
                            ‘</ZONE>’

UTM_LONGITUDE      ::= ‘<UTM_LONGITUDE>’
                             non-negative-integer
                            ‘</UTM_LONGITUDE>’

UTM_LATITUDE       ::= ‘<UTM_LATITUDE>’
                             non-negative-integer
                            ‘</UTM_LATITUDE>’
```

If the map datum is not specified within a **UTM_COORD_LOC**, then “WSG-84” is assumed.

The zone must be between 1 and 60 inclusive

Example:

```
<UTM_COORD_LOC>
    <ALTITUDE>100.7 e 5</ALTITUDE>
    <DATUM>WSG-84</DATUM>
    <ZONE>30</ZONE>
    <UTM_LONGITUDE>704250</UTM_LONGITUDE>
    <UTM_LATITUDE>3391520</UTM_LATITUDE>
</UTM_COORD_LOC>
```

G.4.5.3 Raw GPS

[Editor’s Note: To be completed when GPS defined]

G.4.6 Direction

The type (and sub-elements) are used to specify metadata described in Direction.

```
direction(N)        ::= ‘<’ N timestamp? ‘>’
                          YAW?
                          PITCH?
                          ROLL?
                        ‘</’ N ‘>’
```

```
YAW                 ::= ‘<YAW>’
                          non-negative-real
                        ‘</YAW>’
```

```
PITCH               ::= ‘<PITCH>’
                          real
                        ‘</PITCH>’
```

```
ROLL                ::= ‘<ROLL>’
                          real
                        ‘</ROLL>’
```

The yaw value must be between 0 and 360 inclusive.

The pitch value must be between –90 and 90 inclusive.

The roll value must be between –180 and 180 inclusive.

Example:

```
<YAW>180</YAW>
<PITCH>-10</PITCH>
<ROLL>15</ROLL>
```

G.4.7 Position

The type (and sub-elements) are used to specify metadata described in Position.

```
position(N)         ::= ‘<’ N timestamp? ‘>’
                          (
                               (POINT | RECT) | RECT REGION
                          )?
                          PROPRIETARY?
                          COMMENT?
                        ‘</’ N ‘>’
```

```
POINT               ::= ‘<POINT>’
                          <X>non-negative-integer</X>
                          <Y>non-negative-integer</Y>
                        ‘</POINT>’
```

```
RECT                ::= ‘<RECT>’
                          <X>non-negative-integer</X>
                          <Y>non-negative-integer</Y>
                          <WIDTH>non-negative-integer</WIDTH>
                          <HEIGHT>non-negative-integer</HEIGHT>
                        ‘</RECT>’
```

```
REGION              ::= ‘<REGION>’
                          (
                               <X1>non-negative-integer</X1>
                               <Y1>non-negative-integer</Y1>
                               (
                                    <X2>non-negative-integer</X2>
                                    <Y2>non-negative-integer</Y2>
                                    <X3>non-negative-integer</X3>
                                    <Y3>non-negative-integer</Y3>
```

```
                                   )?
                             )+
                           '</REGION>'
```

Example:

```
<POINT>
    <X>180</X>   <Y>120</Y>
</POINT>
<COMMENT>Not including the tail of the fish</COMMENT>
```

Example:

```
<RECT>
    <X>180</X>  <Y>120</Y>
    <WIDTH>10</WIDTH>  <HEIGHT>10</HEIGHT>
</RECT>
<REGION>
    <X1>180</X1>
    <Y1>120</Y1>
    <X1>190</X1>
    <Y1>120</Y1>
    <X1>190</X1>
    <Y1>130</Y1>
    <X2>180</X2>
    <Y2>130</Y2>
    <X3>180</X3>
    <Y3>130</Y3>
</REGION>
```

G.4.8 Product Details

The type (and sub-elements) are used to specify metadata described in A.2.3.9 Product Details.

```
product_details(N)    ::=    '<' N language? timestamp? '>'
                                  MANUFACTURER?
                                  MODEL?
                                  SERIAL?
                             '</' N '>'
```

```
MANUFACTURER        ::= '<MANUFACTURER>'
                          string
                        '</MANUFACTURER>'
```

```
MODEL               ::= '<MODEL>'
                          string
                        '</MODEL>'
```

```
SERIAL              ::= '<SERIAL>'
                          string
                        '</SERIAL>'
```

Example:

```
<MANUFACTURER>Canon</MANUFACTURER>
<MODEL>EOS 1000F</MODEL>
<SERIAL>2941554</SERIAL>
```

G.5 Fields

G.5.1 Language Identifier

The attribute is used to specify metadata described in A.3.1 Language.

```
language            ::= ‘xml:lang’ ‘=’ ‘“’ string ‘”’
```

The string is of the format described in [RFC1766]. Note that only languages can be specified, and not localities.

Examples:

```
  xml:lang = “en”
  xml:lang = “fr”
```

G.5.2 Timestamp

The attribute is used to specify metadata described in A.3.2 Timestamp.

```
timestamp           ::= ‘timestamp’ ‘=’ ‘“’ digit digit digit digit ‘-’ digit digit ‘-’
                        digit digit (‘T’ digit digit ‘:’ digit digit ‘:’ digit digit (‘.’
                        digit+)?)? ((‘+’ | ‘-’)? digit digit ‘:’ digit digit)? ‘”’
```

Examples:

```
timestamp = “1968-08-19T18:30”
timestamp = “1968-08-19T06:30”
timestamp = “1968-08-19T18:30:10+10:00”
timestamp = “1968-08-19T18:30:10.52”
timestamp = “1968-08-19+10:00”
timestamp = “1968-08”
```

G.5.3 Comment

The element is used to specify metadata described in A.3.3 Comment.

```
COMMENT             ::= ‘<’ ‘COMMENT’ language? timestamp? ‘>’
                          string
                        ‘</COMMENT>’
```

Example:

```
<COMMENT xml:lang=“en” timestamp=“1968-08-19T18:30”>
    I actually caught this fish - honest!
</COMMENT>
```

G.5.4 Proprietary

This element is used for extensibility. The field can contain any sub-field and is used to contain fields in a separate namespace.

The attribute BLIND_WRITE is used to specify that the proprietary information was written to the metadata after the metadata was edited where the authoring application did not understand the proprietary data. If this attribute is omitted, a value of `false` is assumed.

Use of the PROPRIETARY field assumes that DTDs are combined to produce a valid XML document.

Note: It is the intention of DIG35 to define a method to allow such extensions, however, it is yet to be determined if the proposed method is a suitable method at this time.

```
PROPRIETARY         ::= ‘<’ ‘PROPRIETARY’ language?
                                timestamp? BLIND_WRITE? ‘>’
                          ANY
                        ‘</PROPRIETARY>’
```

```
BLIND_WRITE         ::= 'BLIND_WRITE' '=' '"' boolean '"'
```

The following is the example for using the proprietary flag:

```
<?xml version="1.0" ?>
<!DOCTYPE METADATA SYSTEM "dig35.dtd"
  [
    <!ELEMENT WEIGHT (#PCDATA)>
  ]
>
<METADATA>
  <CONTENT>
    <PERSON>
      <PERSON_NAME>
        <NC>Craig</NC>
        <NC TYPE="family">Brown</NC>
      </PERSON_NAME>
      <PROPRIETARY>
        <WEIGHT>75</WEIGHT>
      </PROPRIETARY>
    </PERSON>
  </CONTENT>
</METADATA>
```

G.5.5 URI Locator

The attribute is used to specify a URI reference of an object. See [Xlink].

```
uri-loc             ::= 'xlink:href' '=' '"' uri '"'
```

Examples:

```
 xlink:href = "http:www.digitalimaging.org"
 xlink:href = "image.jpg"
```

[Editor's note: Need to further investigate Xlink specification.]

Annex H: DIG35 XML Syntax Definition

H.1 Overview

This annex specifies the XML encoding for the DIG35 metadata defined in Annex B-F. It defines the mapping between the fields in the definitions and the elements and attributes in XML.

A Document Type Definition (DTD) is also specified to define this mapping (See Annex I). The DTD is not complex enough to define all constraints on the XML encoding of the metadata, and thus the DTD along with this document should be used to ensure valid metadata is created.

When XML Schema becomes a standard, a XML Schema for metadata will also be defined.

H.2 Basic Image Parameter

The element is used to specify metadata described in Annex B: Basic Image Parameter Metadata.

```
BASIC_PARAM         ::= ‘<’ ‘BASIC_PARAM’ language? timestamp?>’
                          BASIC_INFO?
                          OUTPUT?
                          COLOR_INFO?
                          CHANNEL_INFO?
                        ‘</BASIC_PARAM>’
```

H.2.1 Basic Image Information

The element is used to specify metadata described in Basic Image Information.

```
BASIC_INFO          ::= ‘<’ ‘BASIC_INFO’ language? timestamp? ‘>’
                          FILE?
                          IMAGE_ID?
                          IMAGE_SIZE?
                          BPS*
                          NUM_COMPONENTS?
                          COMPRESSION?
                        ‘</BASIC_INFO>’
```

H.2.1.1 File and Format

The element is used to specify metadata described in File and Format.

```
FILE                ::= ‘<’ ‘FILE’ uri-loc? timestamp? ‘/>’
```

File format and version can be specified using an XML notation.

Example:

```
<!DOCTYPE METADATA PUBLIC “dig35_metadata” “…” [
<!NOTATION PDF PUBLIC
   “-//IETF//NONSGML Media Type application/pdf/EN”
   “http://www.isi.edu/in-notes/iana/assignments/media-   types/application/pdf”>
<!ENTITY image_file SYSTEM “image_files/fish.pdf” NDATA PDF>
]>
…
<FILE_FORMAT>
    <FILE_NAME REF=“image_file” />
    <FORMAT>jpeg</FORMAT>
    <FORMAT_VERSION>1</FORMAT_VERSION>
</FILE_FORMAT>
```

H.2.1.2 Image Identifier

The element is used to specify metadata described in Image Identifier.

```
IMAGE_ID            ::= '<' 'IMAGE_ID' language? timestamp? '>'
                          string
                        '</IMAGE_ID>'
```

H.2.1.3 Image Size

The element is used to specify metadata described in Image Size.

```
IMAGE_SIZE          ::= '<' 'IMAGE_SIZE' timestamp? '>'
                          '<WIDTH>'
                               positive-integer
                          '</WIDTH>'
                          '<HEIGHT>'
                               positive-integer
                          '</HEIGHT>'
                        '</IMAGE_SIZE>'
```

Example:

```
<IMAGE_SIZE>
    <WIDTH>100</WIDTH>
    <HEIGHT>100</HEIGHT>
</IMAGE_SIZE>
```

H.2.1.4 Bits Per Component

The element is used to specify metadata described in Bits per components.

```
BPS                 ::= '<' 'BPC' timestamp? '>'
                          positive-integer
                        '</BPC>'
```

The order of the BPS elements within the BASIC_INFO element is important. The order within the metadata matches the order of the components in the image data.

Example:

```
<BPC>8</BPC>
<BPC>8</BPC>
<BPC>8</BPC>
<BPC>1</BPC>
```

H.2.1.5 Number of Components

The element is used to specify metadata described in Number of Components.

```
NUM_COMPONENTS      ::= '<' 'NUM_COMPONENTS' timestamp? '>'
                          positive-integer
                        '</NUM_COMPONENTS>'
```

Example:

```
<NUM_COMPONENTS>4</NUM_COMPONENTS>
```

H.2.1.6 Compression Method

The element is used to specify metadata described in Compression Method.

```
COMPRESSION         ::= '<' 'COMPRESSION' language? timestamp? '>'
                          string
                        '</COMPRESSION>'
```

Example:

```
< COMPRESSION>JPEG</COMPRESSION>
```

H.2.2 Preferred Output Parameter

The element is used to specify metadata described in Preferred Output Parameter.

```
OUTPUT              ::= '<' 'OUTPUT' timestamp? '>'
                          '<WIDTH>'
                               non-negative-real
                          '</WIDTH>'
                          '<HEIGHT>'
                               non-negative-real
                          '</HEIGHT>'
                        '</OUTPUT>'
```

Example:

```
<OUTPUT>
    <WIDTH>0.1524</WIDTH>
    <HEIGHT>0.1016</HEIGHT>
</OUTPUT>
```

H.2.3 Color Information

The element is used to specify metadata described in Color Information.

```
COLOR_INFO          ::= '<' 'COLOR_INFO' language? timestamp? '>'
                          COLORSPACE+
                        '</COLOR_INFO>'
```

```
COLORSPACE          ::= '<' 'COLORSPACE' language? timestamp? uri? '>'
                          string?
                        '</COLORSPACE>'
```

Example:

```
<COLOR_INFO>
    <COLORSPACE REF="color_space_1" />
    <COLORSPACE>sRGB</COLORSPACE>
</COLOR_INFO>
```

H.2.4 Channel Information

The element is used to specify metadata described in Channel Information.

```
CHANNEL_INFO        ::= '<' 'CHANNEL_INFO'
                          language? timestamp? PREMULTIPLIED? '>'
                          CHANNEL+
                        '</CHANNEL_INFO>'
PREMULTIPLIED       ::= 'PREMULTIPLIED' '=' '"' boolean '"'
CHANNEL             ::= '<CHANNEL>'
                          CH_NAME
                          SIZE
                        '</CHANNEL>'
```

```
CH_NAME             ::= '<CH_NAME>'
                          string
                        '</CH_NAME>'
```

```
SIZE                ::= '<SIZE>'
                          string
                        '</SIZE>'
```

The number of channels is implicitly defined by the number of `CHANNEL` sub-elements specified within `CHANNEL_INFO`.

The order of the `CHANNEL` elements within the CHANNEL_INFO element is important. The order within the metadata matches the order of the components in the image data.

Example:

```
<CHANNEL_INFO PREMULTIPLIED="true">
    <CHANNEL>
        <NAME>red</NAME>
        <SIZE>8</SIZE>
    </CHANNEL>
    <CHANNEL>
        <NAME>green</NAME>
        <SIZE>8</SIZE>
    </CHANNEL>
    <CHANNEL>
        <NAME>blue</NAME>
        <SIZE>8</SIZE>
    </CHANNEL>
</CHANNEL_INFO>
```

H.3 Image Creation

The element is used to specify metadata described in Annex C: Image Creation Metadata.

```
CREATION             ::= '<' 'CREATION' language? timestamp? '>'
                           GENERAL_CREATION?
                           CAMERA_CAPTURE?
                           SCANNER_CAPTURE?
                           CAPTURED_ITEM?
                         '</CREATION>'
```

H.3.1 General Information

The element is used to specify metadata described in General Creation Information.

```
GENERAL_CREATION     ::= '<' 'GENERAL_CREATION' language? timestamp? '>'
                           CAPTURE_TIME?
                           SOURCE?
                           SCENE_TYPE?
                           IMAGE_CREATOR?
                           OPERATOR_ORG?
                           OPERATOR_ID?
                         '</GENERAL_CREATION>'

CAPTURE_TIME         ::= date-time ('CAPTURE_TIME')

SOURCE               ::= '<' 'SOURCE' language? timestamp? '>'
                           string
                         '</SOURCE>'

SCENE_TYPE           ::= '<' 'SCENE_TYPE' language? timestamp? '>'
                           string
                         '</SCENE_TYPE>'

IMAGE_CREATOR        ::= '<' 'IMAGE_CREATOR' language? timestamp? '>'
                           string
                         '</IMAGE_CREATOR>'

OPERATOR_ORG         ::= ORGANIZATION ('OPERATOR_ORG')

OPERATOR_ID          ::= '<' 'OPERATOR_ID' language? timestamp? '>'
                           string
                         '</OPERATOR_ID>'
```

Example:

```
<GENERAL_CREATION>
    <CAPTURE_TIME>
        <YEAR>1999</YEAR>
        <MONTH>12</MONTH>
        <DAY>9</DAY>
        <HOUR>12</HOUR>
        <MINUTE>30</MINUTE>
        <SECOND>31</SECOND>
    </CAPTURE_TIME>
    <SOURCE>Digital camera</SOURCE>
    <SCENE_TYPE>Original scene</SCENE_TYPE>
    <IMAGE_CREATOR>Kats Ishii</IMAGE_CREATOR>
</GENERAL_CREATION>
```

H.3.2 Camera Capture Metadata

The element is used to specify metadata described in Camera Capture Metadata.

```
CAMERA_CAPTURE     ::= '<' 'CAMERA_CAPTURE' language? timestamp? '>'
                         CAMERA_INFO?
                         SOFTWARE_INFO?
                         LENS_INFO?
                         DEVICE?
                         CAMERA_SETTINGS?
                         ACCESSORY*
                       '</CAMERA_CAPTURE>'
```

```
CAMERA_INFO        ::= product_details('CAMERA_INFO')
```

```
SOFTWARE_INFO      ::= product_details('SOFTWARE_INFO')
```

```
LENS_INFO          ::= product_details('LENS_INFO')
```

```
ACCESSORY          ::= product_details('ACCESSORY')
```

Example:

```
<CAMERA_CAPTURE>
    <CAMERA_INFO>
        <MANUFACTURER>Canon</MANUFACTURER>
        <MODEL>Canon PowerShot S10</MODEL>
    </CAMERA_INFO>
</CAMERA_CAPTURE>
```

Example:

```
<GENERAL_CREATION>
    <CAMERA_INFO>
        <MANUFACTURER>Canon</MANUFACTURER>
        <MODEL>EOS 1000F</MODEL>
        <SERIAL>2941554</SERIAL>
    </CAMERA_INFO>
    <SOFTWARE_INFO>…</SOFTWARE_INFO>
    <LENS_INFO>…</LENS_INFO>
    <DEVICE>…</DEVICE>
    <CAMERA_SETTINGS>…</CAMERA_SETTINGS>
    <ACCESSORY>
        Remote trigger
    </ACCESSORY>
    <ACCESSORY>
        Tripod
    </ACCESSORY>
</GENERAL_CREATION>
```

H.3.2.1 Device Characterization

The element is used to specify metadata described in Device Characterization Metadata.

```
DEVICE             ::= '<' 'DEVICE' language? timestamp? '>'
                         SENSOR_TECHNOLOGY?
                         FOCAL_PLANE_RES?
                         SPECTRAL_SENSITIVITY?
                         ISO_SATURATION?
                         ISO_NOISE?
                         SPATIAL_FREQ_RESP?
                         CFA_PATTERN?
                         OECF?
                         MAX_APERTURE?
                       '</DEICE>'
```

```
SENSOR_TECHNOLOGY ::= '<' 'SENSOR_TECHNOLOGY' language? timestamp? '>'
```

```
                              string
                            '</SENSOR_TECHNOLOGY>'

FOCAL_PLANE_RES       ::= '<' 'FOCAL_PLANE_RES' language? timestamp? '>'
                              <X>non-negative-integer</X>
                              <Y>non-negative-integer</Y>
                            '</FOCAL_PLANE_RES>'

SPECTRAL_SENSITIVITY  ::= '<SPECTRAL_SENSITIVITY>'
                              string
                            '</SPECTRAL_SENSITIVITY>'

ISO_SATURATION        ::= '<ISO_SATURATION>'
                              decimal
                            '</ISO_SATURATION>'

ISO_NOISE             ::= '<ISO_NOISE>'
                              decimal
                            '</ISO_NOISE>'

SPATIAL_FREQ_RESP     ::= '<SPATIAL_FREQ_RESP>'
                              SPATIAL_FREQ_VAL+
                            '</SPATIAL_FREQ_RESP>'

SPATIAL_FREQ_VAL      ::= '<SPATIAL_FREQ_VAL>'
                              '<SPATIAL_FREQ>'
                                   non-negative-real
                              '</SPATIAL_FREQ>'
                              '<HORIZONTAL_SFR>'
                                   non-negative-real
                              '</HORIZONTAL_SFR>'
                              '<VERTICAL_SFR>'
                                   non-negative-real
                              '</VERTICAL_SFR>'
                            '</SPATIAL_FREQ_VAL>'

CFA_PATTERN           ::= '<CFA_PATTERN>'
                              COLOR_ROW+
                            '</CFA_PATTERN>'

COLOR_ROW             ::= '<COLOR_ROW>'
                              COLOR+
                            '</COLOR_ROW>'

COLOR                 ::= '<COLOR>'
                              string
                            '</COLOR>'

OECF                  ::= '<OECF>'
                              LOG_VAL+
                            '</OECF>'

LOG_VAL               ::= '<LOG_VAL>'
                              LOG_EXPOSURE
                              OUTPUT_LEVEL+
                            '</LOG_VAL>'

LOG_EXPOSURE          ::= '<LOG_EXPOSURE>'
                              real
                            '</LOG_EXPOSURE>'

OUTPUT_LEVEL          ::= '<OUTPUT_LEVEL>'
                              non-negative-real
                            '</OUTPUT_LEVEL>'

MAX_APERTURE          ::= '<MAX_APERTURE>'
                              non-negative-real
                            '</MAX_APERTURE>'
```

The number of color sub elements in each color row must be identical.

Example:

```
<DEVICE>
    <SENSOR_TECHNOLOGY>
        Color sequential linear sensor
    </SENSOR_TECHNOLOGY>
    <SPATIAL_FREQ_RESP>
        <SPATIAL_FREQ_VAL>
            <SPATIAL_FREQ>0.1</SPATIAL_FREQ>
            <HORIZONTAL_SFR>1.00</HORIZONTAL_SFR>
            <VERTICAL_SFR>1.00</VERTICAL_SFR>
        </SPATIAL_FREQ_VAL>
        <SPATIAL_FREQ_VAL>
            <SPATIAL_FREQ>0.2</SPATIAL_FREQ>
            <HORIZONTAL_SFR>0.90</HORIZONTAL_SFR>
            <VERTICAL_SFR>0.95</VERTICAL_SFR>
        </SPATIAL_FREQ_VAL>
        <SPATIAL_FREQ_VAL>
            <SPATIAL_FREQ>0.3</SPATIAL_FREQ>
            <HORIZONTAL_SFR>0.80</HORIZONTAL_SFR>
            <VERTICAL_SFR>0.85</VERTICAL_SFR>
        </SPATIAL_FREQ_VAL>
    </SPATIAL_FREQ_RESP>
    <CFA_PATTERN>
        <COLOR_ROW>
            <COLOR>Green</COLOR>
            <COLOR>Red</COLOR>
            <COLOR>Green</COLOR>
            <COLOR>Red</COLOR>
        </COLOR_ROW>
        <COLOR_ROW>
            <COLOR>Blue</COLOR>
            <COLOR>Green</COLOR>
            <COLOR>Blue</COLOR>
            <COLOR>Green</COLOR>
        </COLOR_ROW>
        <COLOR_ROW>
            <COLOR>Green</COLOR>
            <COLOR>Red</COLOR>
            <COLOR>Green</COLOR>
            <COLOR>Red</COLOR>
        </COLOR_ROW>
        <COLOR_ROW>
            <COLOR>Blue</COLOR>
            <COLOR>Green</COLOR>
            <COLOR>Blue</COLOR>
            <COLOR>Green</COLOR>
        </COLOR_ROW>
    </CFA_PATTERN>
    <OECF>
        <LOG_VAL>
            <LOG_EXPOSURE>-3.0</LOG_EXPOSURE>
            <OUTPUT_LEVEL>10.2</OUTPUT_LEVEL>
            <OUTPUT_LEVEL>12.5</OUTPUT_LEVEL>
            <OUTPUT_LEVEL>8.9</OUTPUT_LEVEL>
        </LOG_VAL>
        <LOG_VAL>
            <LOG_EXPOSURE>-2.0</LOG_EXPOSURE>
            <OUTPUT_LEVEL>48.1</OUTPUT_LEVEL>
            <OUTPUT_LEVEL>47.5</OUTPUT_LEVEL>
            <OUTPUT_LEVEL>48.3</OUTPUT_LEVEL>
        </LOG_VAL>
        <LOG_VAL>
```

```
            <LOG_EXPOSURE>-1.0</LOG_EXPOSURE>
            <OUTPUT_LEVEL>150.2</OUTPUT_LEVEL>
            <OUTPUT_LEVEL>152.0</OUTPUT_LEVEL>
            <OUTPUT_LEVEL>149.8</OUTPUT_LEVEL>
        </LOG_VAL>
    </OECF>
    <MAX_APERTURE>
        10
    </MAX_APERTURE>
</DEVICE>
```

H.3.3 Camera Settings

The element is used to specify metadata described in Camera Settings.

```
CAMERA_SETTINGS   ::= '<' 'CAMERA_SETTINGS' language? timestamp?>'
                        EXP_TIME?
                        SHUTTER_SPEED?
                        F_NUMBER?
                        APERTURE?
                        EXP_PROGRAM?
                        BRIGHTNESS?
                        EXPOSURE_BIAS?
                        SUBJECT_DISTANCE?
                        METERING_MODE?
                        SCENE_ILLUMINANT?
                        COLOR_TEMP?
                        FOCAL_LENGTH?
                        FLASH?
                        FLASH_ENERGY?
                        FLASH_RETURN?
                        BACK_LIGHT?
                        SUBJECT_LOCATION?
                        EXPOSURE_INDEX?
                        AUTO_FOCUS?
                        SPECIAL_EFFECTS*
                        CAMERA_LOCATION?
                        ORIENTATION?
                        PAR?
                      '</CAMERA_SETTINGS>'

EXP_TIME          ::= '<EXP_TIME>'
                        non-negative-real | real
                      '</EXP_TIME>'

SHUTTER_SPEED     ::= '<SHUTTER_SPEED>'
                        real
                      '</SHUTTER_SPEED>'

F_NUMBER          ::= '<F_NUMBER>'
                        non-negative-real
                      '</F_NUMBER>'

APERTURE          ::= '<APERTURE>'
                        real
                      '</APERTURE>'

EXP_PROGRAM       ::= '<EXP_PROGRAM>'
                        string
                      '</EXP_PROGRAM>'

BRIGHTNESS        ::= '<BRIGHTNESS>'
                        real
                      '</BRIGHTNESS>'

EXPOSURE_BIAS     ::= '<EXPOSURE_BIAS>'
```

```
                            real
                          '</EXPOSURE_BIAS>'

SUBJECT_DISTANCE    ::=   '<SUBJECT_DISTANCE>'
                            real
                          '</SUBJECT_DISTANCE>'

METERING_MODE       ::=   '<METERING_MODE>'
                            string
                          '</METERING_MODE>'

SCENE_ILLUMINANT    ::=   '<SCENE_ILLUMINANT>'
                            string
                          '</SCENE_ILLUMINANT>'

COLOR_TEMP          ::=   '<COLOR_TEMP>'
                            non-negative-real
                          '</COLOR_TEMP>'

FOCAL_LENGTH        ::=   '<FOCAL_LENGTH>'
                            non-negative-real
                          '</FOCAL_LENGTH>'

FLASH               ::=   '<FLASH>'
                            boolean
                          '</FLASH>'

FLASH_ENERGY        ::=   '<FLASH_ENERGY>'
                            non-negative-real
                          '</FLASH_ENERGY>'

FLASH_RETURN        ::=   '<FLASH_RETURN>'
                            boolean
                          '</FLASH_RETURN>'

BACK_LIGHT          ::=   '<BACK_LIGHT>'
                            string
                          '</BACK_LIGHT>'

SUBJECT_LOCATION    ::=   location('SUBJECT_LOCATION')

EXPOSURE_INDEX      ::=   '<EXPOSURE_INDEX>'
                            real
                          '</EXPOSURE_INDEX>'

AUTO_FOCUS          ::=   '<AUTO_FOCUS>'
                            string
                          '</AUTO_FOCUS>'

SPECIAL_EFFECTS     ::=   '<SPECIAL_EFFECTS>'
                            string
                          '</SPECIAL_EFFECTS>'

CAMERA_LOCATION     ::=   location('CAMERA_LOCATION')

ORIENTATION         ::=   direction('ORIENTATION')

PAR                 ::=   '<PAR>'
                            '<WIDTH>'
                                 non-negative-real
                            '</WIDTH>'
                            '<HEIGHT>'
                                 non-negative-real
                            '</HEIGHT>'
                          '</PAR>'
```

Example:

```
<CAMERA_SETTINGS>
    <EXP_TIME> … </EXP_TIME>
    <SHUTTER_SPEED> … </SHUTTER_SPEED>
    <F_NUMBER> … </F_NUMBER>
    <APERTURE> … </APERTURE>
    <EXP_PROGRAM> … </EXP_PROGRAM>
    <BRIGHTNESS> … </BRIGHTNESS>
    <EXPOSURE_BIAS> … </EXPOSURE_BIAS>
    <SUBJECT_DISTANCE> … </SUBJECT_DISTANCE>
    <METERING_MODE> … </METERING_MODE>
    <SCENE_ILLUMINANT> … </SCENE_ILLUMINANT>
    <COLOR_TEMP> … </COLOR_TEMP>
    <FOCAL_LENGTH> … </FOCAL_LENGTH>
    <FLASH> … </FLASH>
    <FLASH_ENERGY> … </FLASH_ENERGY>
    <FLASH_RETURN> … </FLASH_RETURN>
    <BACK_LIGHT> … </BACK_LIGHT>
    <SUBJECT_LOCATION> … </SUBJECT_LOCATION>
    <EXPOSURE_INDEX> … </EXPOSURE_INDEX>
    <AUTO_FOCUS> … </AUTO_FOCUS>
    <SPECIAL_EFFECTS> … </SPECIAL_EFFECTS>
    <SPECIAL_EFFECTS> … </SPECIAL_EFFECTS>
    <CAMERA_LOCATION> … </CAMERA_LOCATION>
    <ORIENTATION> … </ORIENTATION>
    <PAR>
        <WIDTH>1.0,/WIDTH>
        <HEIGHT>3.2</HEIGHT>
    </PAR>
</CAMERA_SETTINGS>
```

H.3.4 Scanner Capture Metadata

The element is used to specify metadata described in Scanner Capture Metadata.

```
SCANNER_CAPTURE    ::= '<' 'SCANNER_CAPTURE' language? timestamp? '>'
                         SCANNER_INFO?
                         SOFTWARE_INFO?
                         SCANNER_INFO   ::= product_details('SCANNER_INFO')
SOFTWARE_INFO      ::= product_details('SOFTWARE_INFO')
SCANNER_SETTINGS?
                       '</SCANNER_CAPTURE>'

SCANNER_INFO       ::= product_details('SCANNER_INFO')

SOFTWARE_INFO      ::= product_details('SOFTWARE_INFO')

SCANNER_SETTINGS   ::= '<' 'SCANNER_SETTINGS' language? timestamp? '>'
                         PIXEL_SIZE?
                         PHYSICAL_SCAN_RES?
                       '</SCANNER_SETTINGS>'

PIXEL_SIZE         ::= '<PIXEL_SIZE>'
                         real
                       '</PIXEL_SIZE>'

PHYSICAL_SCAN_RES ::= '<PHYSICAL_SCAN_RES>'
                         <X>real</X>
                         <Y>real</Y>
                       '</PHYSICAL_SCAN_RES>'
```

Example:

```
<SCANNER_CAPTURE>
    <SCANNER_INFO>
        <MANUFACTURER>Agfa</MANUFACTURER>
        <MODEL>SnapScan Touch</MODEL>
    </SCANNER_INFO>
</SCANNER_CAPTURE>
```

H.3.5 Captured Item Metadata

The element is used to specify metadata described in Captured Item Metadata.

```
CAPTURED_ITEM      ::= ‘<’ ‘CAPTURED_ITEM’ language? timestamp? ‘>’
                         REFLECTION_PRINT?
                         FILM?
                       ‘</CAPTURED_ITEM>’
```

H.3.5.1 Reflection Print

The element is used to specify metadata described in Captured Item Metadata.

```
REFLECTION_PRINT   ::= ‘<’ ‘REFLECTION_PRINT’ language? timestamp? ‘>’
                         DOCUMENT_SIZE?
                         MEDIUM?
                         RP_TYPE?
                       ‘</REFLECTION_PRINT>’
```

```
DOCUMENT_SIZE      ::= ‘<DOCUMENT_SIZE>’
                         <X>real</X>
                         <Y>real</Y>
                       ‘</DOCUMENT_SIZE>’
```

```
MEDIUM             ::= ‘<MEDIUM>’
                         string
                       ‘</MEDIUM>’
```

```
RP_TYPE            ::= ‘<RP_TYPE>’
                         string
                       ‘</RP_TYPE>’
```

H.3.5.2 Film

The element is used to specify metadata described in Film.

```
FILM               ::= ‘<’ ‘FILM’ language? timestamp? ‘>’
                         BRAND?
                         CATEGORY?
                         FILM_SIZE?
                         ROLL_ID?
                         FRAME_ID?
                         FILM_SPEED?
                       ‘</FILM>’
```

```
BRAND              ::= ‘<BRAND>’
                         string
                       ‘</BRAND>’
```

```
CATEGORY           ::= ‘<CATEGORY>’
                         string
                       ‘</CATEGORY>’
```

```
FILM_SIZE          ::= ‘<FILM_SIZE>’
                         <X>real</X>
                         <Y>real</Y>
                       ‘</FILM_SIZE>’
```

```
ROLL_ID            ::= ‘<ROLL_ID>’
                         positive-integer
                       ‘</ROLL_ID>’
```

```
FRAME_ID           ::= ‘<FRAME_ID>’
                         positive-integer
                       ‘</FRAME_ID>’
```

```
FILM_SPEED         ::= ‘<FILM_SPEED>’
                         string
                       ‘</FIM_SPEED>’
```

H.4 Content Description

The element is used to specify metadata described in Annex D: Content Description Metadata.

```
CONTENT              ::= '<' 'CONTENT' language? timestamp?>'
                           ROLL_CAPTION?
                           CAPTION?
                           CAPTURE_TIME?
                           PERSON*
                           THING*
                           ORGANIZATION*
                           KEYWORD*
                           CONTENT_LOCATION?
                           EVENT*
                           AUDIO*
                           PROPRIETARY?
                           COMMENT?
                         '</CONTENT>'

ROLL_CAPTION         ::= '<' 'ROLL_CAPTION' language? timestamp? '>'
                           string
                         '</ROLL_CAPTION>'

CAPTION              ::= '<' 'CAPTION' language? timestamp? '>'
                           string
                         '</CAPTION>'

CAPTURE_TIME         ::= date-time ('CAPTURE_TIME')

CONTENT_LOCATION     ::= location ('CONTENT_LOCATION')
loc_pos_kw (N)       ::= location('N_LOCATION')?
                         position('N_POSITION')?
                         KEYWORD*
```

H.4.1 Person Description

The element is used to specify metadata described in Person Description.

```
PERSON               ::=  person ('PERSON', loc_pos_kw (PERSON) )
```

H.4.2 Organization Description

The element is used to specify metadata described in Organization Description.

```
ORGANIZATION         ::= organization ('ORGANIZATION', loc_pos_kw (ORG))
```

H.4.3 Thing

The element is used to specify metadata described in Thing.

```
THING                ::= '<' 'THING' language? timestamp? '>'
                           PROPRIETARY?
                           COMMENT
                           THING_POSITION?
                           THING_LOCATION?
                           THING*
                         '</THING>'

THING_POSITION       ::= position ('THING_POSITION')

THING_LOCATION       ::= location ('THING_LOCATION')
```

H.4.4 Keyword Set / Keyword

The element is used to specify metadata described in Keyword Set / Keyword.

```
KEYWORD              ::= ‘<’ ‘KEYWORD’ language? timestamp?
                                          (‘DICTIONARY’ ‘=’ string)? ‘>’
                           (
                                string?
                                (
                                     (
                                          PROPRIETARY?
                                          COMMENT?
                                     )
                                     | KEYWORD*
                                )
                         )
                         ‘</KEYWORD>’
```

H.4.5 Event

The element is used to specify metadata described in Event.

```
EVENT                ::= ‘<’ ‘EVENT’ language? timestamp? ‘>’
                           EVENT_TYPE
                           PARTICIPANT*
                           EVENT_TIME?
                           DURATION?
                           LOCATION?
                           PROPRIETARY?
                           COMMENT?
                           EVENT*
                         ‘</EVENT>’
```

```
EVENT_TYPE           ::= ‘<’ ‘EVENT_TYPE’ language? ‘>’
                           string
                         ‘</EVENT_TYPE>’
```

```
PARTICIPANT          ::= ‘<’ ‘PARTICIPANT’ language?
                                                     (‘REF’ ‘=’ ‘IDREF’)? ‘>’
                           string
                         ‘</PARTICIPANT>’
```

```
EVENT_TIME           ::= date-time (‘EVENT_TIME’)
```

```
DURATION             ::= date-time (‘DURATION’)
```

```
LOCATION             ::= location(‘LOCATION’)
```

H.4.6 Audio

The element is used to specify metadata described in Audio.

```
AUDIO                ::= ‘<’ ‘AUDIO’ language? timestamp? uri-loc? ‘>’
                           PROPRIETARY?
                           COMMENT?
                         ‘</AUDIO>’
```

H.5 History

The element is used to specify metadata described in Annex E: History Metadata.

```
HISTORY              ::= '<' 'HISTORY' language? timestamp?>'
                           IMG_PROCESSING?
                           METADATA_HISTORY?
                         '</HISTORY>'
```

H.5.1 Image Processing

The element is used to specify metadata described in Image Processing Hints Metadata.

```
IMG_PROCESSING           ::= '<' 'IMG_PROCESSING' language? timestamp? '>'
                                IMG_CREATED?
                                IMG_CROPPED?
                                IMG_ROTATED?
                                IMG_GTC_ADJUSTED?
                                IMG_SPACIALLY_ADJUSTED?
                                IMG_RETOUCHED?
                                IMG_EXTENSIVELY_EDITED?
                                IMG_COMPOSITED?
                                IMG_STC_ADJUSTED?
                           '</IMG_PROCESSING>'
```

```
IMG_CREATED               ::= '<IMG_CREATED/>'
```

```
IMG_CROPPED               ::= '<IMG_CROPPED/>'
```

```
IMG_ROTATED               ::= '<IMG_ROTATED/>'
```

```
IMG_GTC_ADJUSTED          ::= '<IMG_GTC_ADJUSTED/>'
```

```
IMG_SPACIALLY_ADJUSTED    ::= '<IMG_SPACIALLY_ADJUSTED/>'
```

```
IMG_RETOUCHED             ::= '<IMG_RETOUCHED/>'
```

```
IMG_EXTENSIVELY_EDITED    ::= '<IMG_EXTENSIVELY_EDITED/>'
```

```
IMG_COMPOSITED            ::= '<IMG_COMPOSITED/>'
```

```
IMG_STC_ADJUSTED          ::= '<IMG_STC_ADJUSTED/>'
```

Example: A cropped and retouched image

```
<IMG_PROCESSING>
    <IMG_CREATED />
    <IMG_CROPPED />
    <IMG_RETOUCHED />
</IMG_PROCESSING>
```

Example: A highly processed image

```
<IMG_PROCESSING>
    <IMG_CREATED />
    <IMG_CROPPED />
    <IMG_ROTATED />
    <IMG_GTC_ADJUSTED />
    <IMG_SPACIALLY_ADJUSTED />
    <IMG_EXTENSIVELY_EDITED />
    <IMG_COMPOSITED />
</IMG_PROCESSING>
```

H.5.2 Image Metadata History

The element is used to specify metadata described in Image Metadata History.

```
METADATA_HISTORY  ::= '<' 'METADATA_HISTORY' language? timestamp? '>'
                        UIID? |
                        (
                            BASIC_PARAM?
                            CREATION?
                            CONTENT?
                            IPR?
                        ) |
                        METADATA_HISTORY*
                      '</METADATA_HISTORY>'
```

```
UIID              ::= '<' 'UIID' language? timestamp? '>'
                        string
                      '</UIID>'
```

Example: Metadata history referring to a UIID

```
<METADATA_HISTORY>
    <UIID>
        <!-- a unique ID -->
    </UIID>
</METADATA_HISTORY>
```

Example: History of the previous metadata after image processing (e.g. cropping)

```
<METADATA_HISTORY>
    <CREATION>
        <!-- original creation metadata (e.g. camera settings) -->
    </CREATION>
    <CONTENT>
        <!-- original content description metadata (e.g. people) -->
    </CONTENT>
</METADATA_HISTORY>
```

Example: Recursive History

```
<METADATA_HISTORY>
    < METADATA_HISTORY >
        <UIID>
            <!-- a unique ID -->
        </UIID>
    </METADATA_HISTORY >
</METADATA_HISTORY>
```

H.6 Intellectual Property Rights

The element is used to specify metadata described in Annex F: IPR Metadata.

```
IPR                  ::= '<' 'IPR' language? timestamp?>'
                          IPR_NAMES?
                          IPR_DESCRIPTION?
                          IPR_DATES?
                          IPR_EXPLOITATION?
                          IPR_IDENTIFICATION?
                          IPR_CONTACT_POINT?
                       '</IPR>'
```

H.6.1 Names

The element is used to specify metadata described in Names.

```
IPR_NAMES              ::= '<' 'IPR_NAMES' language? timestamp? '>'
                               IPR_ORIGINAL_AUTHOR?
                               IPR_IMAGE_CREATOR?
                               IPR_RIGHT_HOLDER?
                               COPYRIGHT?
                            '</IPR_NAMES>'
```

```
IPR_ORIGINAL_AUTHOR    ::= '<' 'IPR_ORIGINAL_AUTHOR' language? timestamp? '>'
                               string
                            '</IPR_ORIGINAL_AUTHOR>'
```

```
IPR_IMAGE_CREATOR      ::= '<' 'IPR_IMAGE_CREATOR' language? timestamp? '>'
                               string
                            '</IPR_IMAGE_CREATOR>'
```

```
IPR_RIGHT_HOLDER       ::= '<' 'IPR_RIGHT_HOLDER' language? timestamp? '>'
                               string
                            '</IPR_RIGHT_HOLDER>'
```

H.6.2 Description

The element is used to specify metadata described in Description.

```
IPR_DESCRIPTION      ::= '<' 'IPR_DESCRIPTION' language? timestamp? '>'
                          IPR_TITLE?
                          IPR_LEGEND?
                          IPR_CAPTION?
                          COPYRIGHT?
                       '</IPR_DESCRIPTION>'
```

```
IPR_TITLE            ::= '<' 'IPR_TITLE' language? timestamp? '>'
                          string
                       '</IPR_TITLE>'
```

```
IPR_LEGEND           ::= '<' 'IPR_LEGEND' language? timestamp? '>'
                          string
                       '</IPR_LEGEND>'
```

```
IPR_CAPTION          ::= '<' 'IPR_CAPTION' language? timestamp? '>'
                          string
                       '</IPR_CAPTION>'
```

```
COPYRIGHT            ::= '<' 'COPYRIGHT' language? timestamp? '>'
                          string
                       '</COPYRIGHT>'
```

H.6.3 Dates

The element is used to specify metadata described in Dates.

```
IPR_DATES           ::= ‘<’ ‘IPR_DATES’ language? timestamp? ‘>’
                          ORIGINAL_CREATION_DATE?
                          PICTURE_TAKEN_DATE?
                          SCAN_DATE?
                          PROCESSING_DATE?
                          MODIFICATION_DATE?
                          LAST_MODIFICATION_DATE?
                        ‘</IPR_DATES>’
```

```
ORIGINAL_CREATION_DATE  ::= date-time(‘ORIGINAL_CREATION_DATE’)
```

```
PICTURE_TAKEN_DATE      ::= date-time(‘PICTURE_TAKEN_DATE_DATE’)
```

```
SCAN_DATE               ::= date-time(‘SCAN_DATE’)
```

```
PROCESSING_DATE         ::= date-time(‘PROCESSING_DATE’)
```

```
MODIFICATION_DATE       ::= date-time(‘MODIFICATION_DATE’)
```

```
LAST_MODIFICATION_DATE  ::= date-time(‘LAST_MODIFICATION_DATE’)
```

H.6.4 Exploitation

The element is used to specify metadata described in Exploitation.

```
IPR_EXPLOITATION    ::= ‘<’ ‘IPR_EXPLOITATION’ language? timestamp? ‘>’
                          IPR_PROTECTION?
                          IPR_RESTRICTIONS_OF_USE?
                          IPR_OBLIGATIONS?
                          IPR_MGMT_SYS?
                        ‘</IPR_EXPLOITATION>’
```

```
IPR_PROTECTION          ::= ‘<’ ‘IPR_PROTECTION’ timestamp? ‘>’
                                 string
                              ‘</IPR_PROTECTION>’
```

```
IPR_RESTRICTIONS_OF_USE ::= ‘<’ ‘IPR_ RESTRICTIONS_OF_USE’
                                              language? timestamp? ‘>’
                                 string
                               ‘</IPR_RESTRICTIONS_OF_USE>’
```

```
IPR_OBLIGATIONS         ::= ‘<’ ‘IPR_OBLIGATIONS’ language? timestamp? ‘>’
                                 string
                              ‘</IPR_OBLIGATIONS>’
```

H.6.4.1 IPR Management System

The element is used to specify metadata described in Exploitation.

```
IPR_MGMT_SYS        ::= ‘<’ ‘IPR_MGMT_SYS’ language? timestamp? ‘>’
                          IPR_MGMT_TYPE?
                          IPR_MGMT_SYS_ID?
                          IPR_MGMT_SYS_LOCATION?
                        ‘</IPR_MGMT_SYS>’
```

```
IPR_MGMT_TYPE       ::= ‘<IPR_MGMT_TYPE>’
                          string
                        ‘</IPR_MGMT_TYPE>’
```

```
IPR_MGMT_SYS_ID     ::= ‘<IPR_MGMT_SYS_ID>’
                          string
                        ‘</IPR_MGMT_SYS_ID>’
```

```
IPR_MGMT_SYS_LOCATION ::= ‘<’ ‘IPR_MGMT_SYS_LOCATION’ uri-loc ‘/>’
```

H.6.5 Identification

The element is used to specify metadata described in Identification.

```
IPR_IDENTIFICATION    ::= '<' 'IPR_IDENTIFICATION'
                                                 language? timestamp? '>'
                            IPR_IDENTIFIER?
                            LICENCE_PLATE?
                          '</IPR_IDENTIFICATION>'
```

H.6.5.1 Generic IPR Identifier

The element is used to specify metadata described in Generic IPR Identifier.

```
IPR_IDENTIFIER      ::= '<' 'IPR_IDENTIFIER' timestamp? '>'
                            IPR_ID_MODE?
                            IPR_ID?
                          '</IPR_IDENTIFIER>'
```

```
IPR_ID_MODE         ::= '<IPR_ID_MODE>'
                            string
                          '</IPR_ID_MODE>'
```

```
IPR_ID              ::= '<IPR_ID>'
                            string
                          '</IPR_ID>'
```

H.6.5.2 License Plate

The element is used to specify metadata described in License Plate.

```
LICENCE_PLATE       ::= '<' 'LICENCE_PLATE'  language? timestamp? '>'
                            LP_COUNTRY
                            LP_REG_AUT
                            LP_REG_NUM
                          '</LICENCE_PLATE>'
```

```
LP_COUNTRY          ::= '<LP_COUNTRY>'
                            identifier
                          '</LP_COUNTRY>'
```

```
LP_REG_AUT          ::= '<LP_REG_AUT>'
                            integer
                          '</LP_REG_AUT>'
```

```
LP_REG_NUM          ::= '<LP_REG_NUM>'
                            integer
                          '</LP_REG_NUM>'
```

H.6.6 Contact Point

The element is used to specify metadata described in Contact Point.

```
IPR_CONTACT_POINT ::= '<' 'IPR_CONTACT_POINT' language? timestamp? '>'
                            IPR_ADDRESS?
                            IPR_LINK?
                            IPR_COLLECTION?
                          '</IPR_CONTACT_POINT>'
```

```
IPR_ADDRESS         ::= '<IPR_ADDRESS>'
                            string
                          '</IPR_ADDRESS>'
```

```
IPR_LINK            ::= '<IPR_LINK>'
                            string
                          '</IPR_LINK>'
```

```
IPR_COLLECTION    ::= '<' 'IPR_COLLECTION' uri-loc '>'
```

Annex I: Complete DIG35 XML DTD

```
<!-- ********************************************************************* -->
<!-- DIG35 Metadata DTD 2000-03-06                                         -->
<!--                                                                       -->
<!-- This version is for DIG35 Metadata Working Draft 1.0 (20000306)       -->
<!-- ********************************************************************* -->

<!-- Predefined General Entities -->
<!ENTITY  lt                              "&#38;#60;">
<!ENTITY  gt                              "&#62;">
<!ENTITY  amp                             "&#38;#38;">
<!ENTITY  apos                            "&#39;">
<!ENTITY  quot                            "&#34;">

<!-- ===================================================================== -->
<!-- DIG35 Root element                                                    -->
<!-- ===================================================================== -->
<!ELEMENT METADATA                       ((BASIC_PARAM?,
                                           CREATION?,
                                           CONTENT?,
                                           HISTORY?,
                                           IPR?),
                                          METADATA*) >
<!ATTLIST METADATA                        xml:lang CDATA "en"
                                          TYPE (SINGLE | COLLECTION) "SINGLE" >

<!-- ===================================================================== -->
<!-- Fundamental Type and Field Definitions                                -->
<!-- ===================================================================== -->

<!-- Attribute definitions -->

<!ENTITY % att-lang                        "xml:lang CDATA #IMPLIED" >
<!ENTITY % att-timestamp                   "TIMESTAMP CDATA #IMPLIED" >
<!ENTITY % att-lang-ts                     "%att-lang; %att-timestamp;" >
<!ENTITY % att-lang-ts-id                  "%att-lang-ts; IDENTIFIER ID #IMPLIED" >
<!ENTITY % att-uri-loc
 "xmlns:xlink                              CDATA #FIXED 'http://www.w3.org/XML/XLink'
  xlink:href                               CDATA #REQUIRED" >

<!-- Date and Time Type -->

<!ENTITY % date-element                   "YEAR?, MONTH?, DAY?, SEASON?" >
<!ENTITY % time-element                   "HOUR?, MINUTE?, SECOND?" >
<!ENTITY % dig35-date                     "(%date-element;, PROPRIETARY?,
                                           COMMENT?)" >
<!ENTITY % dig35-time                     "(%time-element;, PROPRIETARY?,
                                           COMMENT?)" >
<!ENTITY % dig35-date-time                "(%date-element;, %time-element;,
                                          PROPRIETARY?, COMMENT?)" >
<!ELEMENT DATE_TIME                       %dig35-date-time; >
<!ATTLIST DATE_TIME                       %att-lang-ts; >

<!ELEMENT DATE                            (%date-element;, SEASON?,
                                          PROPRIETARY?, COMMENT?)>
<!ATTLIST DATE                            %att-lang-ts; >

<!ELEMENT TIME                            (%time-element;,
                                          PROPRIETARY?, COMMENT?) >
<!ATTLIST TIME                            %att-lang-ts; >

<!ELEMENT YEAR                            (#PCDATA)>
<!ELEMENT MONTH                           (#PCDATA)>
<!ELEMENT DAY                             (#PCDATA)>
```

```
<!ELEMENT SEASON                        (#PCDATA)>
<!ATTLIST SEASON                        %att-lang;>

<!ELEMENT HOUR                          (#PCDATA)>
<!ELEMENT MINUTE                        (#PCDATA)>
<!ELEMENT SECOND                        (#PCDATA)>


<!-- Product Description Type-->

<!ENTITY % dig35-product-desc           "(MANUFACTURER?, MODEL?, SERIAL? )" >

<!ELEMENT MANUFACTURER                  (#PCDATA)>
<!ELEMENT MODEL                         (#PCDATA)>
<!ELEMENT SERIAL                        (#PCDATA)>

<!-- Person Description Type-->

<!ENTITY % dig35-person                 "(NAME_TITLE?,
                                         PERSON_NAME*, NICK_NAME?,
                                         JOB_TITLE?, COMPANY_NAME?,
                                         ADDRESS*, PHONE*, EMAIL*, WEB*,
                                         BIRTH_DATE?, AGE?,
                                         PROPRIETARY?, COMMENT?)" >

<!ELEMENT NAME_TITLE                    (#PCDATA)>
<!ATTLIST NAME_TITLE                    %att-lang-ts; >

<!ELEMENT PERSON_NAME                   (NC+)>
<!ATTLIST PERSON_NAME                   %att-lang-ts; >

<!ELEMENT NC                            (#PCDATA)>
<!ATTLIST NC                            TYPE (prefix | given | family |
                                              suffix) "given" >

<!ELEMENT NICK_NAME                     (#PCDATA)>
<!ATTLIST NICK_NAME                     %att-lang-ts; >

<!ELEMENT JOB_TITLE                     (#PCDATA)>
<!ATTLIST JOB_TITLE                     %att-lang-ts; >

<!ELEMENT COMPANY_NAME                  (#PCDATA)>
<!ATTLIST COMPANY_NAME                  %att-lang-ts; >

<!ELEMENT PHONE                         (#PCDATA)>
<!ATTLIST PHONE                         %att-timestamp;
                                        TYPE CDATA #IMPLIED >

<!ELEMENT EMAIL                         (#PCDATA)>
<!ATTLIST EMAIL                         %att-lang-ts;
                                        TYPE CDATA #IMPLIED >

<!ELEMENT WEB                           EMPTY>
<!ATTLIST WEB                           %att-lang-ts;
                                        %att-uri-loc;
                                        TYPE CDATA #IMPLIED >

<!ELEMENT BIRTH_DATE                    %dig35-date; >

<!ELEMENT AGE                           (#PCDATA)>
<!ATTLIST AGE                           %att-lang-ts; >

<!-- Organization type -->
<!ENTITY % dig35-org                    "(ORGANIZATION_NAME?,
                                         ADDRESS*, PHONE*, EMAIL*, WEB*,
                                         PROPRIETARY?, COMMENT?)" >

<!ELEMENT ORGANIZATION_NAME             (#PCDATA)>
<!ATTLIST ORGANIZATION_NAME             %att-lang-ts; >
```

```
<!-- Comment field -->

<!ELEMENT COMMENT                           (#PCDATA)>
<!ATTLIST COMMENT                           %att-lang-ts; >


<!-- Proprietary field -->

<!ELEMENT PROPRIETARY                       ANY>
<!ATTLIST PROPRIETARY                       %att-lang-ts;
                                            BLIND_WRTIE (TRUE | FALSE) "FALSE" >


<!-- Location type -->

<!ENTITY % dig35-location                  "(COORD_LOC?, UTM_COORD_LOC?, ADDRESS?,
                                            GPS?, PROPRIETARY?, COMMENT?)" >
<!ELEMENT LOCATION                          %dig35-location; >
<!ATTLIST LOCATION                          %att-lang-ts; >

<!ELEMENT COORD_LOC                        (LONGITUDE?, LATITUDE?) >
<!ATTLIST COORD_LOC                         %att-timestamp; >

<!ELEMENT LONGITUDE                         (#PCDATA)>
<!ELEMENT LATITUDE                          (#PCDATA)>

<!ELEMENT UTM_COORD_LOC                    (ALTITUDE?, DATUM?, ZONE?,
                                            UTM_LONGITUDE?, UTM_LATITUDE?) >
<!ATTLIST UTM_COORD_LOC                     %att-timestamp; >

<!ELEMENT ALTITUDE                          (#PCDATA)>
<!ELEMENT DATUM                             (#PCDATA)>
<!ELEMENT ZONE                              (#PCDATA)>
<!ELEMENT UTM_LONGITUDE                     (#PCDATA)>
<!ELEMENT UTM_LATITUDE                      (#PCDATA)>



<!--  Address type -->

<!ELEMENT ADDRESS                          (ADDR_NAME?, ADDR*,
                                            (POSTCODE | ZIPCODE)?,
                                            COUNTRY?, TYPE?) >
<!ATTLIST ADDRESS                           %att-lang-ts; >

<!ELEMENT ADDR_NAME                        (#PCDATA)>

<!ELEMENT ADDR                             (#PCDATA)>
<!ATTLIST ADDR                             TYPE CDATA #IMPLIED>

<!ELEMENT POSTCODE                         (#PCDATA)>
<!ELEMENT ZIPCODE                          (#PCDATA)>

<!ELEMENT COUNTRY                          (#PCDATA)>
<!ATTLIST COUNTRY                          %att-lang;>

<!ELEMENT TYPE                             (#PCDATA)>


<!-- GPS type -->
<!-- This section still needs completion depending on DIG35 specification changes. -->

<!ELEMENT GPS                              (COORD_LOC?, TIME?, GPS_SATELLITES?,
                                            GPS_STATUS?, GPS_MEASURE_MODE?,
                                            GPS_DOP?, GPS_SPEED?, GPS_TRACK?,
                                            GPS_IMAGE_DIR?, GPS_MAP_DATUM?,
                                            GPS_DEST_LOC?, GPS_DEST_BEARING?,
                                            GPS_DEST_DISTANCE?) >

<!ELEMENT GPS_SATELLITES                   (#PCDATA)>
<!ELEMENT GPS_STATUS                       (#PCDATA)>
```

```
<!ELEMENT GPS_MEASURE_MODE                (#PCDATA)>
<!ELEMENT GPS_DOP                         (#PCDATA)>
<!ELEMENT GPS_SPEED                       (#PCDATA)>
<!ELEMENT GPS_TRACK                       (#PCDATA)>
<!ELEMENT GPS_IMAGE_DIR                   (#PCDATA)>
<!ELEMENT GPS_MAP_DATUM                   (#PCDATA)>
<!ELEMENT GPS_DEST_LOC                    (ZONE, LONGITUDE, LATITUDE) >
<!ELEMENT GPS_DEST_BEARING                (#PCDATA)>
<!ELEMENT GPS_DEST_DISTANCE               (#PCDATA)>

<!-- Direction type-->

<!ENTITY % dig35-direction                "(YAW?, PITCH?, ROLL?)" >
<!ELEMENT DIRECTION                       %dig35-direction; >
<!ATTLIST DIRECTION                       %att-timestamp; >

<!ELEMENT YAW                             (#PCDATA)>
<!ELEMENT PITCH                           (#PCDATA)>
<!ELEMENT ROLL                            (#PCDATA)>


<!-- Position type -->
<!-- Changed; need clarification -->
<!ENTITY % dig35-position                "(((POINT | RECT) |
                                           RECT | REGION)?,
                                           PROPRIETARY?, COMMENT?)" >
<!ELEMENT POSITION                         %dig35-position; >
<!ATTLIST POSITION                         %att-lang-ts; >

<!ELEMENT POINT                            (X, Y)>
<!ELEMENT RECT                             (X, Y, WIDTH, HEIGHT)>
<!ELEMENT REGION                           ((X1, Y1, X2?, Y2?, X3?, Y3?)*)>

<!ELEMENT X                               (#PCDATA)>
<!ELEMENT Y                               (#PCDATA)>
<!ELEMENT WIDTH                           (#PCDATA)>
<!ELEMENT HEIGHT                          (#PCDATA)>
<!ELEMENT X1                              (#PCDATA)>
<!ELEMENT Y1                              (#PCDATA)>
<!ELEMENT X2                              (#PCDATA)>
<!ELEMENT Y2                              (#PCDATA)>
<!ELEMENT X3                              (#PCDATA)>
<!ELEMENT Y3                              (#PCDATA)>


<!-- ======================================================================== -->
<!-- Basic Image Parameter                                                    -->
<!-- ======================================================================== -->

<!ELEMENT BASIC_PARAM                     (BASIC_INFO?, OUTPUT?,
                                           COLOR_INFO?, CHANNEL_LIST?) >
<!ATTLIST BASIC_PARAM                      %att-lang-ts; >


<!ELEMENT BASIC_INFO                      (FILE?, IMAGE_ID?, IMAGE_SIZE?,
                                           BITS_PER_COMP*, NUM_COMPONENTS?,
                                           COMPRESSION?) >
<!ATTLIST BASIC_INFO                       %att-lang-ts; >

<!ELEMENT FILE                             EMPTY>
<!ATTLIST FILE                             REF ENTITY #IMPLIED >

<!ELEMENT IMAGE_ID                         (#PCDATA)>
<!ATTLIST IMAGE_ID                         %att-lang-ts; >

<!ELEMENT IMAGE_SIZE                       (WIDTH, HEIGHT)>
<!ATTLIST IMAGE_SIZE                       %att-timestamp; >

<!ELEMENT BITS_PER_COMP                    (#PCDATA)>
<!ATTLIST BITS_PER_COMP                    %att-timestamp; >
```

```
<!ELEMENT NUM_COMPONENTS                  (#PCDATA)>
<!ATTLIST NUM_COMPONENTS                  %att-timestamp; >

<!ELEMENT COMPRESSION                     (#PCDATA)>
<!ATTLIST COMPRESSION                     %att-lang-ts; >


<!ELEMENT OUTPUT                          (WIDTH, HEIGHT) >
<!ATTLIST OUTPUT                          %att-timestamp; >


<!ELEMENT COLOR_INFO                      (COLORSPACE+) >
<!ATTLIST COLOR_INFO                      %att-lang-ts; >

<!ELEMENT COLORSPACE (#PCDATA)>
<!ATTLIST COLORSPACE                      %att-lang-ts;
                                          REF ENTITY #IMPLIED >

<!ELEMENT CHANNEL_LIST (CHANNEL+)>
<!ATTLIST CHANNEL_LIST                    %att-lang-ts;
                                          PREMULTIPLIED (TRUE | FALSE) "FALSE" >

<!ELEMENT CHANNEL                         (CH_NAME, SIZE) >

<!ELEMENT CH_NAME                         (#PCDATA)>
<!ELEMENT SIZE                            (#PCDATA)>


<!-- ======================================================================= -->
<!-- Image Creation                                                          -->
<!-- ======================================================================= -->

<!ELEMENT CREATION                       (GENERAL_CREATION?,
                                          CAMERA_CAPTURE?,
                                          SCANNER_CAPTURE?,
                                          CAPTURED_ITEM?) >
<!ATTLIST CREATION                        %att-lang-ts; >

<!-- General Image Creation -->

<!ELEMENT GENERAL_CREATION               (CREATION_TIME?, SOURCE?,
                                          SCENE_TYPE?, IMAGE_CREATOR?,
                                          OPERATOR_ORG?, OPERATOR_ID?) >
<!ATTLIST GENERAL_CREATION                %att-lang-ts; >

<!ELEMENT CREATION_TIME                   %dig35-date-time;>
<!ATTLIST CREATION_TIME                   %att-lang-ts;>

<!ELEMENT SOURCE                          (#PCDATA)>
<!ATTLIST SOURCE                          %att-lang-ts; >

<!ELEMENT SCENE_TYPE                      (#PCDATA)>
<!ATTLIST SCENE_TYPE                      %att-lang-ts; >

<!ELEMENT IMAGE_CREATOR                   (#PCDATA)>
<!ATTLIST IMAGE_CREATOR                   %att-lang-ts; >

<!ELEMENT OPERATOR_ORG                    %dig35-org; >
<!ATTLIST OPERATOR_ORG                    %att-lang-ts; >

<!ELEMENT OPERATOR_ID                     (#PCDATA)>
<!ATTLIST OPERATOR_ID                     %att-lang-ts; >

<!-- Camera capture -->

<!ELEMENT CAMERA_CAPTURE                 (CAMERA_INFO?, SOFTWARE_INFO?,
                                          LENS_INFO?, DEVICE?,
                                          CAMERA_SETTINGS?, ACCESSORY*) >
<!ATTLIST CAMERA_CAPTURE                  %att-lang-ts; >

<!ELEMENT CAMERA_INFO                     %dig35-product-desc;>
```

```
<!ATTLIST CAMERA_INFO                        %att-lang-ts;>

<!ELEMENT SOFTWARE_INFO                      %dig35-product-desc;>
<!ATTLIST SOFTWARE_INFO                      %att-lang-ts;>

<!ELEMENT LENS_INFO                          %dig35-product-desc;>
<!ATTLIST LENS_INFO                          %att-lang-ts;>

<!ELEMENT DEVICE                            (SENSOR_TECHNOLOGY?,
                                             FOCAL_PLANE_RES?,
                                             SPECTRAL_SENSITIVITY?,
                                             ISO_SATURATION?, ISO_NOISE?,
                                             SPATIAL_FREQ_RESP?, CFA_PATTERN?,
                                             OECF?, MAX_APERTURE?) >
<!ATTLIST DEVICE                             %att-lang-ts; >

<!ELEMENT SENSOR_TECHNOLOGY                  (#PCDATA)>
<!ATTLIST SENSOR_TECHNOLOGY                  %att-lang-ts;>

<!ELEMENT FOCAL_PLANE_RES                    (X, Y) >
<!ATTLIST FOCAL_PLANE_RES                    %att-lang-ts;>

<!ELEMENT SPECTRAL_SENSITIVITY               (#PCDATA)>
<!ELEMENT ISO_SATURATION                     (#PCDATA)>
<!ELEMENT ISO_NOISE                          (#PCDATA)>

<!ELEMENT SPATIAL_FREQ_RESP                  (SPATIAL_FREQ_VAL+)>
<!ELEMENT SPATIAL_FREQ_VAL                   (SPATIAL_FREQ,
                                              HORIZONTAL_SFR,
                                              VERTICAL_SFR) >
<!ELEMENT SPATIAL_FREQ                       (#PCDATA)>
<!ELEMENT HORIZONTAL_SFR                     (#PCDATA)>
<!ELEMENT VERTICAL_SFR                       (#PCDATA)>

<!ELEMENT CFA_PATTERN                        (COLOR_ROW+) >
<!ELEMENT COLOR_ROW                          (COLOR+) >
<!ELEMENT COLOR                              (#PCDATA)>

<!ELEMENT OECF                               (LOG_VAL+)>
<!ELEMENT LOG_VAL                            (LOG_EXPOSURE, OUTPUT_LEVEL+)>
<!ELEMENT LOG_EXPOSURE                       (#PCDATA)>
<!ELEMENT OUTPUT_LEVEL                       (#PCDATA)>

<!ELEMENT MAX_APERTURE                       (#PCDATA)>

<!-- Camera settings -->

<!ELEMENT CAMERA_SETTINGS                   (EXP_TIME?, SHUTTER_SPEED?,
                                             F_NUMBER?, APERTURE?,
                                             EXP_PROGRAM?, BRIGHTNESS?,
                                             EXPOSURE_BIAS?, SUBJECT_DISTANCE?,
                                             METERING_MODE?, SCENE_ILLUMINANT?,
                                             COLOR_TEMP?, FOCAL_LENGTH?,
                                             FLASH?, FLASH_ENERGY?, FLASH_RETURN?,
                                             BACK_LIGHT?, SUBJECT_LOCATION?,
                                             EXPOSURE_INDEX?, AUTO_FOCUS?,
                                             SPECIAL_EFFECTS*, CAMERA_LOCATION?,
                                             ORIENTATION?, PAR?)>
<!ATTLIST CAMERA_SETTINGS                    %att-lang-ts; >

<!ELEMENT EXP_TIME                           (#PCDATA)>
<!ELEMENT SHUTTER_SPEED                      (#PCDATA)>
<!ELEMENT F_NUMBER                           (#PCDATA)>
<!ELEMENT APERTURE                           (#PCDATA)>
<!ELEMENT EXP_PROGRAM                        (#PCDATA)>
<!ELEMENT BRIGHTNESS                         (#PCDATA)>
<!ELEMENT EXPOSURE_BIAS                      (#PCDATA)>
<!ELEMENT SUBJECT_DISTANCE                   (#PCDATA)>
<!ELEMENT METERING_MODE                      (#PCDATA)>
<!ELEMENT SCENE_ILLUMINANT                   (#PCDATA)>
<!ELEMENT COLOR_TEMP                         (#PCDATA)>
```

```
<!ELEMENT FOCAL_LENGTH                      (#PCDATA)>
<!ELEMENT FLASH                             (#PCDATA)>
<!ELEMENT FLASH_ENERGY                      (#PCDATA)>
<!ELEMENT FLASH_RETURN                      (#PCDATA)>
<!ELEMENT BACK_LIGHT                        (#PCDATA)>
<!ELEMENT SUBJECT_LOCATION                  %dig35-location; >
<!ELEMENT EXPOSURE_INDEX                    (#PCDATA)>
<!ELEMENT AUTO_FOCUS                        (#PCDATA)>
<!ELEMENT SPECIAL_EFFECTS                   (#PCDATA)>
<!ELEMENT CAMERA_LOCATION                   %dig35-location; >
<!ELEMENT ORIENTATION                       %dig35-direction; >
<!ELEMENT PAR                               (WIDTH, HEIGHT) >

<!ELEMENT ACCESSORY                         %dig35-product-desc;>
<!ATTLIST ACCESSORY                         %att-lang-ts; >

<!-- Scanner Capture -->

<!ELEMENT SCANNER_CAPTURE                  (SCANNER_INFO?, SOFTWARE_INFO?,
                                            SCANNER_SETTINGS?) >
<!ATTLIST SCANNER_CAPTURE                   %att-lang-ts; >

<!ELEMENT SCANNER_INFO                      %dig35-product-desc;>
<!ATTLIST SCANNER_INFO                      %att-lang-ts; >

<!ELEMENT SCANNER_SETTINGS                  (PIXEL_SIZE?, PHYSICAL_SCAN_RES?) >
<!ATTLIST SCANNER_SETTINGS                  %att-timestamp; >

<!ELEMENT PIXEL_SIZE                        (#PCDATA)>
<!ELEMENT PHYSICAL_SCAN_RES                 (X, Y) >

<!-- Captured item -->

<!ELEMENT CAPTURED_ITEM                     (REFLECTION_PRINT | FILM) >
<!ATTLIST CAPTURED_ITEM                     %att-lang-ts; >

<!-- Reflection print -->

<!ELEMENT REFLECTION_PRINT                  (DOCUMENT_SIZE?, MEDIUM?, RP_TYPE?) >
<!ATTLIST REFLECTION_PRINT                  %att-lang-ts; >

<!ELEMENT DOCUMENT_SIZE                     (X, Y) >
<!ELEMENT MEDIUM                            (#PCDATA)>
<!ELEMENT RP_TYPE                           (#PCDATA)>

<!-- Film -->

<!ELEMENT FILM                             (BRAND?, CATEGORY?, FILM_SIZE?,
                                            ROLL_ID?, FRAME_ID?, FILM_SPEED?) >
<!ATTLIST FILM                              %att-lang-ts; >

<!ELEMENT BRAND                             (#PCDATA)>
<!ELEMENT CATEGORY                          (#PCDATA)>
<!ELEMENT FILM_SIZE                         (X, Y)>
<!ELEMENT ROLL_ID                           (#PCDATA)>
<!ELEMENT FRAME_ID                          (#PCDATA)>
<!ELEMENT FILM_SPEED                        (#PCDATA)>


<!-- ====================================================================== -->
<!-- Content Description                                                    -->
<!-- ====================================================================== -->

<!ELEMENT CONTENT                          (ROLL_CAPTION?, CAPTION?,
                                            CAPTURE_TIME?,
                                            PERSON*, THING*, ORGANIZATION*,
                                            KEYWORD*, LOCATION?,
                                            EVENT*, AUDIO*,
                                            PROPRIETARY?, COMMENT?) >
<!ATTLIST CONTENT                           %att-lang-ts; >
```

```
<!ELEMENT ROLL_CAPTION                      (#PCDATA)>
<!ATTLIST ROLL_CAPTION                      %att-lang-ts; >

<!ELEMENT CAPTION                           (#PCDATA)>
<!ATTLIST CAPTION                           %att-lang-ts; >

<!ELEMENT CAPTURE_TIME                      %dig35-date-time;>
<!ATTLIST CAPTURE_TIME                      %att-lang-ts;>

<!ELEMENT PERSON                            (%dig35-person;, LOCATION?,
                                             POSITION?, KEYWORD*)>
<!ATTLIST PERSON                            %att-lang-ts-id; >

<!ELEMENT THING                            (PROPRIETARY?, COMMENT, LOCATION?,
                                            POSITION?, THING*) >
<!ATTLIST THING                             %att-lang-ts-id; >

<!ELEMENT ORGANIZATION                      (%dig35-org;, LOCATION?,
                                             POSITION?, KEYWORD*)>
<!ATTLIST ORGANIZATION                      %att-lang-ts-id; >

<!ELEMENT KEYWORD                          (#PCDATA | PROPRIETARY
                                            | COMMENT | KEYWORD)* >
<!ATTLIST KEYWORD                          %att-lang-ts;
                                           DICTIONARY CDATA #IMPLIED >

<!-- Event -->

<!ELEMENT EVENT                            (EVENT_TYPE, PARTICIPANT*,
                                            EVENT_TIME?, DURATION?,
                                            LOCATION?, PROPRIETARY?, COMMENT?,
                                            EVENT*) >
<!ATTLIST EVENT                             %att-lang-ts;
                                            IDENTIFIER ID #IMPLIED >

<!ELEMENT EVENT_TYPE                        (#PCDATA)>
<!ATTLIST EVENT_TYPE                        %att-lang; >

<!ELEMENT PARTICIPANT                       (#PCDATA)>
<!ATTLIST PARTICIPANT                       %att-lang;
                                            REF IDREF #IMPLIED >

<!ELEMENT EVENT_TIME                        %dig35-date-time;>
<!ATTLIST EVENT_TIME                        %att-lang-ts; >

<!ELEMENT DURATION                          %dig35-date-time;>
<!ATTLIST DURATION                          %att-lang-ts; >

<!-- Audio -->

<!ELEMENT AUDIO                             (PROPRIETARY?, COMMENT?)>
<!ATTLIST AUDIO                             %att-lang-ts; %att-uri-loc; >


<!-- ====================================================================== -->
<!-- History                                                                -->
<!-- ====================================================================== -->

<!ELEMENT HISTORY                          (IMG_PROCESSING?,
                                            METADATA_HISTORY*) >
<!ATTLIST HISTORY                           %att-lang-ts; >

<!-- Image processing -->

<!ELEMENT IMG_PROCESSING                   (IMG_CREATED?, IMG_CROPPED?,
                                            IMG_ROTATED?, IMG_GTC_ADJUSTED?,
                                            IMG_SPACIALLY_ADJUSTED?,
                                            IMG_RETOUCHED?,
                                            IMG_EXTENSIVELY_EDITED?,
                                            IMG_COMPOSITED?, IMG_STC_ADJUSTED?) >
```

```
<!ATTLIST IMG_PROCESSING                    %att-timestamp; >

<!ELEMENT IMG_CREATED                       EMPTY>
<!ELEMENT IMG_CROPPED                       EMPTY>
<!ELEMENT IMG_ROTATED                       EMPTY>
<!ELEMENT IMG_GTC_ADJUSTED                  EMPTY>
<!ELEMENT IMG_SPACIALLY_ADJUSTED            EMPTY>
<!ELEMENT IMG_RETOUCHED                     EMPTY>
<!ELEMENT IMG_EXTENSIVELY_EDITED            EMPTY>
<!ELEMENT IMG_COMPOSITED                    EMPTY>
<!ELEMENT IMG_STC_ADJUSTED                  EMPTY>

<!-- Metadata history -->

<!ELEMENT METADATA_HISTORY                 (UIID? |
                                            (BASIC_PARAM?, CREATION?,
                                             CONTENT?, IPR?) |
                                             METADATA_HISTORY*) >
<!ATTLIST HISTORY                           %att-lang-ts; >

<!ELEMENT UIID                              (#PCDATA)>

<!-- ======================================================================= -->
<!-- Intellectual Property Rights                                            -->
<!-- ======================================================================= -->

<!ELEMENT IPR                              (IPR_NAMES?, IPR_DESCRIPTION?,
                                            IPR_DATES?, IPR_EXPLOITATION?,
                                            IPR_IDENTIFICATION?,
                                            IPR_CONTACT_POINT?) >
<!ATTLIST IPR                               %att-lang-ts; >


<!ELEMENT IPR_NAMES                        (IPR_ORIGINAL_AUTHOR?,
                                            IPR_IMAGE_CREATOR?,
                                            IPR_RIGHT_HOLDER?) >
<!ATTLIST IPR_NAMES                         %att-lang-ts; >

<!ELEMENT IPR_ORIGINAL_AUTHOR               (#PCDATA)>
<!ATTLIST IPR_ORIGINAL_AUTHOR               %att-lang-ts; >

<!ELEMENT IPR_IMAGE_CREATOR                 (#PCDATA)>
<!ATTLIST IPR_IMAGE_CREATOR                 %att-lang-ts; >

<!ELEMENT IPR_RIGHT_HOLDER                  (#PCDATA)>
<!ATTLIST IPR_RIGHT_HOLDER                  %att-lang-ts; >

<!-- IPR description -->
<!ELEMENT IPR_DESCRIPTION                  (IPR_TITLE?, IPR_LEGEND?,
                                            IPR_CAPTION?, COPYRIGHT?) >

<!ELEMENT IPR_TITLE                         (#PCDATA)>
<!ATTLIST IPR_TITLE                         %att-lang-ts; >

<!ELEMENT IPR_LEGEND                        (#PCDATA)>
<!ATTLIST IPR_LEGEND                        %att-lang-ts; >

<!ELEMENT IPR_CAPTION                       (#PCDATA)>
<!ATTLIST IPR_CAPTION                       %att-lang-ts; >

<!ELEMENT COPYRIGHT                         (#PCDATA)>
<!ATTLIST COPYRIGHT                         %att-lang-ts; >


<!ELEMENT IPR_DATES                        (ORIGINAL_CREATION_DATE?,
                                            PICTURE_TAKEN_DATE?, SCAN_DATE?,
                                            PROCESSING_DATE?,
                                            MODIFICATION_DATE?,
                                            LAST_MODIFICATION_DATE?) >
<!ATTLIST IPR_DATES                         %att-lang-ts; >
```

```
<!ELEMENT ORIGINAL_CREATION_DATE            %dig35-date-time;>
<!ATTLIST ORIGINAL_CREATION_DATE            %att-lang-ts;>

<!ELEMENT PICTURE_TAKEN_DATE                %dig35-date-time;>
<!ATTLIST PICTURE_TAKEN_DATE                %att-lang-ts;>

<!ELEMENT SCAN_DATE                         %dig35-date-time;>
<!ATTLIST SCAN_DATE                         %att-lang-ts;>

<!ELEMENT PROCESSING_DATE                   %dig35-date-time;>
<!ATTLIST PROCESSING_DATE                   %att-lang-ts;>

<!ELEMENT MODIFICATION_DATE                 %dig35-date-time;>
<!ATTLIST MODIFICATION_DATE                 %att-lang-ts;>

<!ELEMENT LAST_MODIFICATION_DATE            %dig35-date-time;>
<!ATTLIST LAST_MODIFICATION_DATE            %att-lang-ts;>

<!-- IPR exploitation -->

<!ELEMENT IPR_EXPLOITATION                 (IPR_PROTECTION?,
                                            IPR_RESTRICTIONS_OF_USE?,
                                            IPR_OBLIGATIONS?,
                                            IPR_MGMT_SYS?) >
<!ATTLIST IPR_EXPLOITATION                  %att-lang-ts; >

<!ELEMENT IPR_PROTECTION                    (#PCDATA)>
<!ATTLIST IPR_EXPLOITATION                  %att-timestamp; >

<!ELEMENT IPR_RESTRICTIONS_OF_USE           (#PCDATA)>
<!ATTLIST IPR_RESTRICTIONS_OF_USE           %att-lang-ts; >

<!ELEMENT IPR_OBLIGATIONS                   (#PCDATA)>
<!ATTLIST IPR_OBLIGATIONS                   %att-lang-ts; >

<!-- IPR management system -->

<!ELEMENT IPR_MGMT_SYS                     (IPR_MGMT_TYPE?,
                                            IPR_MGMT_SYS_ID?,
                                            IPR_MGMT_SYS_LOCATION) >
<!ATTLIST IPR_MGMT_SYS                      %att-lang-ts; >

<!ELEMENT IPR_MGMT_TYPE                     (#PCDATA)>

<!ELEMENT IPR_MGMT_SYS_ID                   (#PCDATA)>

<!ELEMENT IPR_MGMT_SYS_LOCATION             EMPTY>
<!ATTLIST IPR_MGMT_SYS_LOCATION             %att-uri-loc;>

<!ELEMENT IPR_IDENTIFICATION               (IPR_IDENTIFIER?,
                                            LICENCE_PLATE?) >

<!ELEMENT IPR_IDENTIFIER                    (IPR_ID_MODE?, IPR_ID)>
<!ATTLIST IPR_IDENTIFIER                    %att-lang-ts; >

<!ELEMENT IPR_ID_MODE                       (#PCDATA)>
<!ELEMENT IPR_ID                            (#PCDATA)>

<!ELEMENT LICENCE_PLATE                    (LP_COUNTRY,
                                            LP_REG_AUT,
                                            LP_REG_NUM) >
<!ATTLIST LICENCE_PLATE                     %att-lang-ts; >

<!ELEMENT LP_COUNTRY                        (#PCDATA)>
<!ELEMENT LP_REG_AUT                        (#PCDATA)>
<!ELEMENT LP_REG_NUM                        (#PCDATA)>

<!ELEMENT IPR_CONTACT_POINT                (IPR_ADDRESS?, IPR_LINK?,
                                            IPR_COLLECTION?) >
<!ELEMENT IPR_ADDRESS                       (#PCDATA)>
```

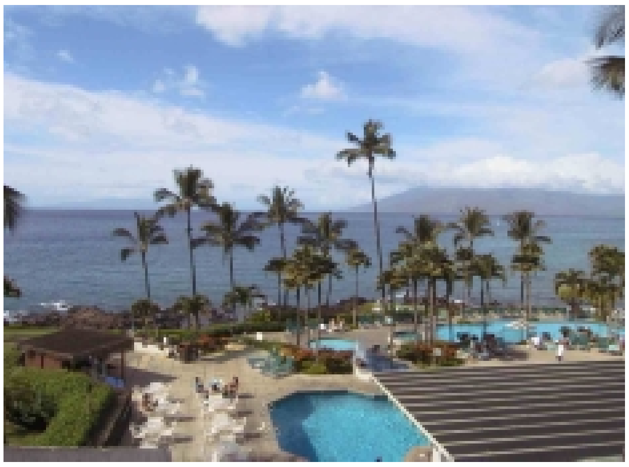
```
<!ELEMENT IPR_LINK                          (#PCDATA)>
<!ELEMENT IPR_COLLECTION                    EMPTY>
<!ATTLIST IPR_MGMT_SYS_LOCATION             %att-uri-loc;>
```

APPENDICIES

I Examples

I.1 Metadata Examples

[Editor's note: Examples to be included with sample image and metadata.]



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE METADATA SYSTEM "dig35-20000306.dtd">

<METADATA>
    <CREATION>
        <GENERAL_CREATION>
            <CREATION_TIME>
                <YEAR>1999</YEAR>
                <MONTH>12</MONTH>
                <DAY>9</DAY>
                <HOUR>12</HOUR>
                <MINUTE>30</MINUTE>
                <SECOND>31</SECOND>
            </CREATION_TIME>
            <IMAGE_CREATOR>Kats Ishii</IMAGE_CREATOR>
        </GENERAL_CREATION>
        <CAMERA_CAPTURE>
            <CAMERA_INFO>
                <MANUFACTURER>XYZ</MANUFACTURER>
                <MODEL>FooBar 100</MODEL>
            </CAMERA_INFO>
            <CAMERA_SETTINGS>
                <EXP_TIME>1/500</EXP_TIME>
                <SHUTTER_SPEED>9</SHUTTER_SPEED>
                <F_NUMBER>5.6</F_NUMBER>
                <APERTURE>5</APERTURE>
                <EXPOSURE_BIAS>0</EXPOSURE_BIAS>
                <SUBJECT_DISTANCE>10</SUBJECT_DISTANCE>
                <FLASH>FALSE</FLASH>
            </CAMERA_SETTINGS>
        </CAMERA_CAPTURE>
    </CREATION>
    <CONTENT>
        <CAPTION>View from Conference Room</CAPTION>
        <LOCATION>
            <ADDRESS>
                <ADDR TYPE="City">Maui</ADDR>
                <COUNTRY>US</COUNTRY>
            </ADDRESS>
        </LOCATION>
        <EVENT>
            <EVENT_TYPE>ISO Meeting</EVENT_TYPE>
        </EVENT>
        <COMMENT>A nice day to swim!?</COMMENT>
    </CONTENT>
</METADATA>
```

I.2 Internal Association Examples

This section discusses several methods to associate XML encoded metadata with various image file formats --- most are currently implemented in digital cameras. Benefits and drawbacks for each method are also addressed.

There are many image file formats commonly use in existing applications. The following table shows a list of file formats and association examples are discussed in this section:

Table I-1: File Format and Association Method Relation

File format (variant)	Association method
Exif/TIFF (Uncompressed)	Using the UserComment Tag
	Using a Private Tag
Exif /JPEG(Compressed)	Using the UserComment Tag
	Using a Private Tag
	Using the APPn marker
TIFF	Using a Private Tag
Flashpix	Using the ContentDescription
	Using the Extension list property set
JPEG2000[i]	Using the XML Box
	Defining DIG35 Boxes
Format independent	Using the end of file

Several alternatives may exist for each file format and the most appropriate method is application dependent. However, if those associated metadata are to be properly exchanged, the sender and the receiver must comply to a common implementation method.

[i] This is based on the Committee Draft dated 1999/12/9.

I.2.1 File Format Dependent Association

I.2.1.1 Exif/TIFF – Use UserComment

Since XML may be multi-byte and extensible, application should store them in the “UserComment” tag defined in the Exif IFD, if to use existing tags.

Note that “MakerNote” and “UserComment” are the only tags that permit multi-byte and arbitrary size data.

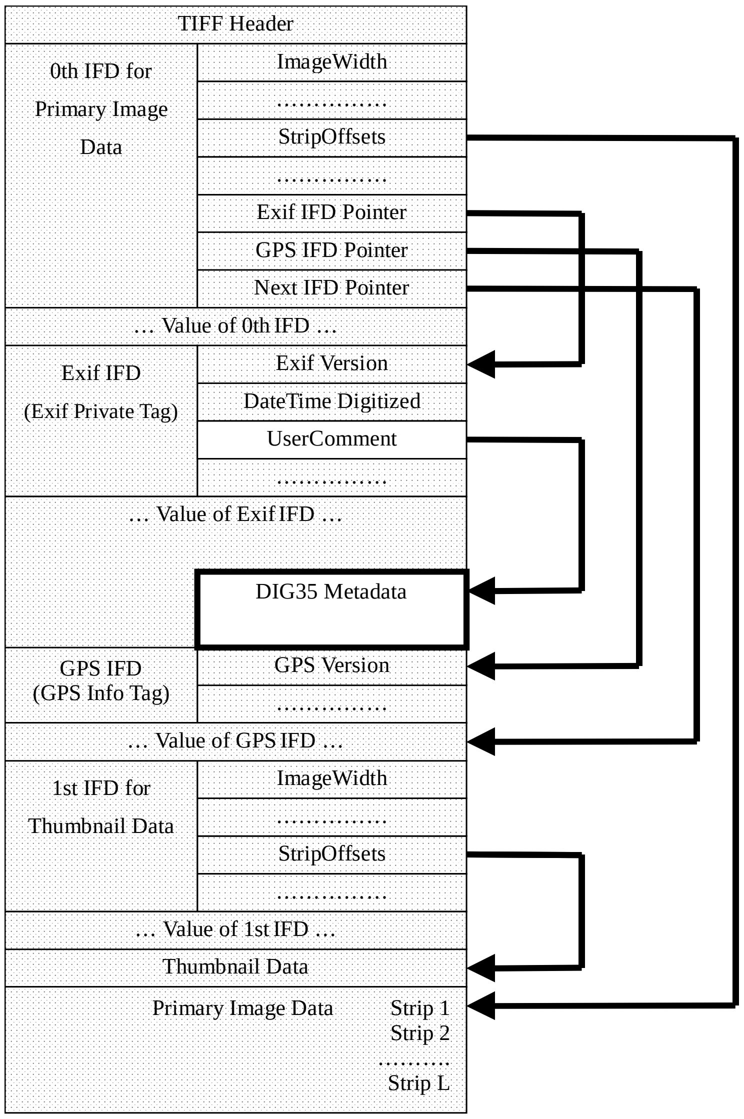


Figure I-1: Uncompressed Exif/TIFF file format with DIG35 metadata

Table I-2: Discussion items on Exif/TIFF using UserComment

Pros	- No need to define a new Tag
Cons	- If other applications use the “UserComment” tag for a different purpose, DIG35 metadata may not use this tag having the risk of being overwritten. - If user increase “UserComment” size, many offset tags need to recalculate each address.

I.2.1.2 Exif/TIFF – Use Private Tag

DIG35 Metadata may be recorded using a "Private Tag" in the 0th IFD.

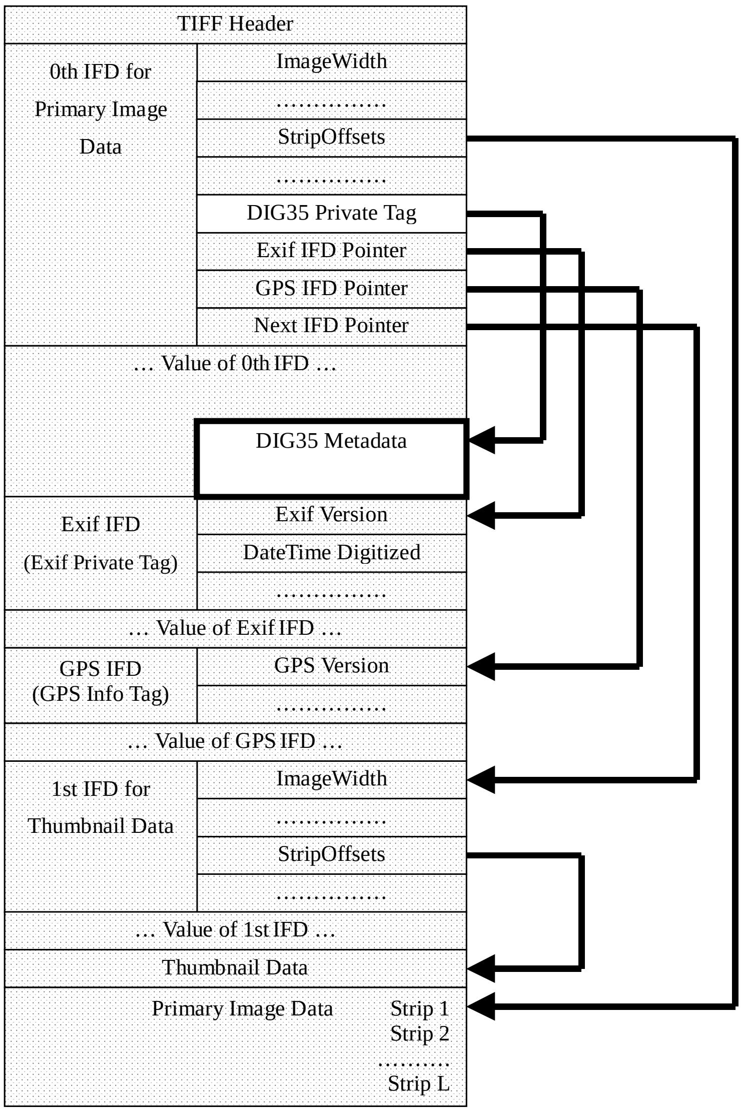


Figure I-2: Uncompressed Exif file format with DIG35 metadata

Table I-3: Discussion items on Exif/TIFF using Private Tag

Pros	- Only DIG35 Metadata uses the new private tag.
Cons	- Need to add a NEW private Tag. - If metadata are added increasing the size, many offset tags need to recalculate each address.

I.2.1.3 Exif/JPEG – Use User Comment Tag

If we record DIG35 information in existing tags, applications must store them in the "UserComment" tag defined in Exif IFD.

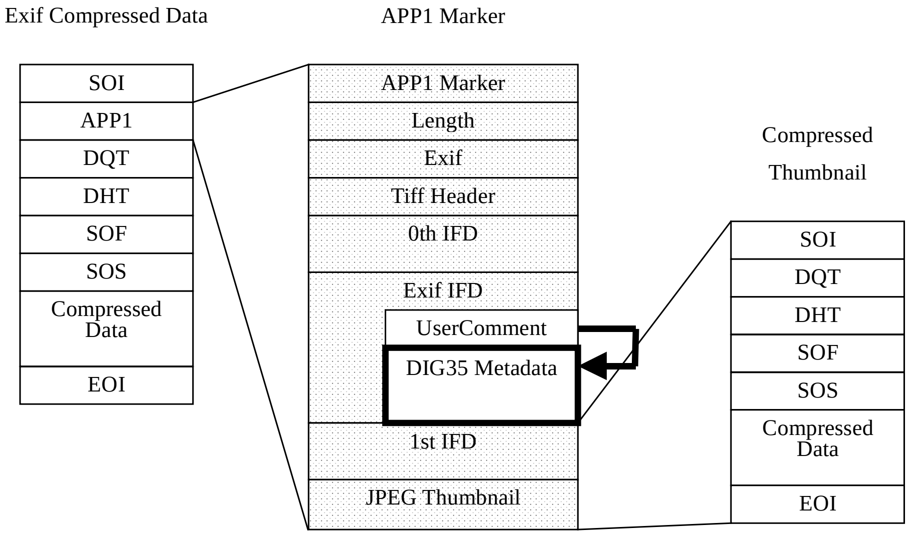


Figure I-3: Compressed Exif/JPEG file format with DIG35 metadata

Table I-4: Discussion items on Exif/JPEG using UserComment

Pros	- No need to define a new Tag
Cons	- If other applications use the "UserComment" tag for a different purpose, DIG35 metadata may not use this tag having the risk of being overwritten. - If user increase "UserComment" size, many offset tags need to recalculate each address.

I.2.1.4 Exif/JPEG – Private Tag Use

DIG35 Metadata may be recorded using a “Private tag” in the 0th IFD.

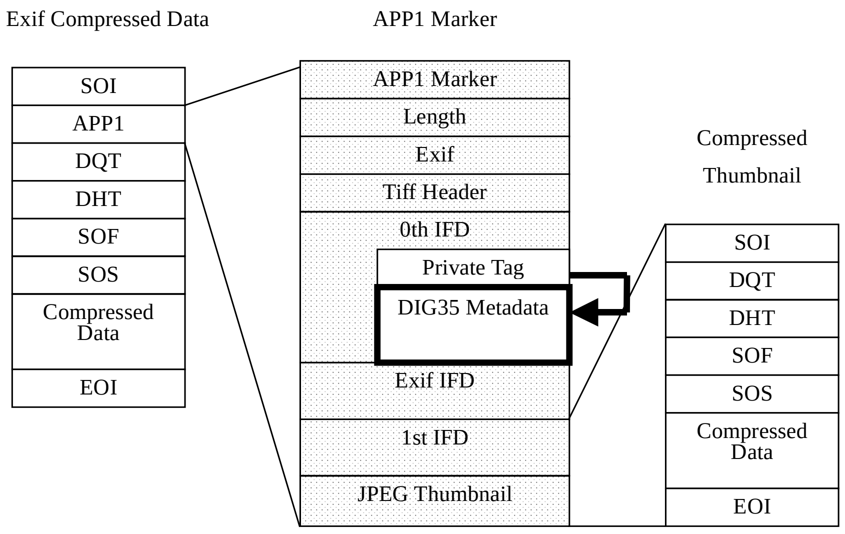


Figure I-4: Compressed Exif data files with DIG35 metadata

Table I-5: Discussion items on Exif/JPEG using Private Tag

Pros	- Only DIG35 Metadata uses the new private tag.
Cons	- Need to add a NEW private Tag. - If metadata are added increasing the size, many offset tags need to recalculate each address.

I.2.1.5 Exif/JPEG – Use APPn marker

DIG35 Metadata may be recorded under a different APP marker. This example uses the APP4 marker.

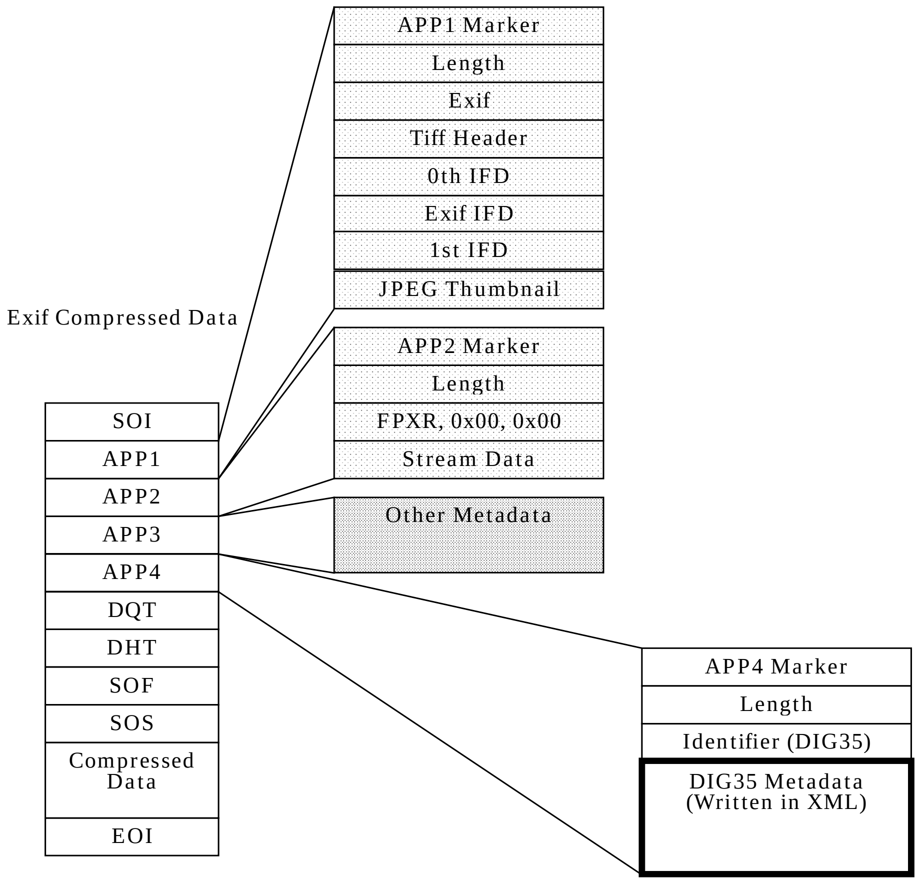


Figure I-5: Compressed Exif data files with DIG35 metadata

Table I-6: Discussion items on Exif/JPEG using Private Tag

PROS	- Only DIG35 Metadata uses the defined APP Marker.
CONS	- Need to communicate the use of the APP Marker. - APP Marker maybe used in other application.

I.2.1.6 TIFF – Private Tag Use

DIG35 Metadata may be recorded using a “Private Tag” in a TIFF file.

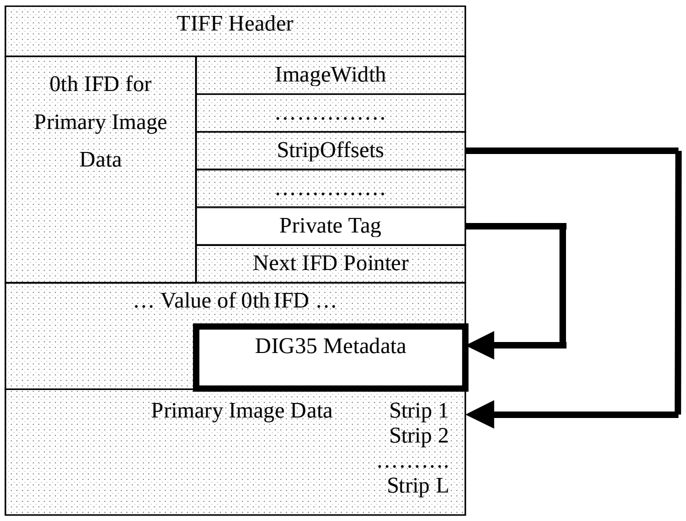


Figure I-6: TIFF file format with DIG35 metadata

Table I-7: Discussion items on TIFF using Private Tag

Pros	- A new private tag is allocated specifically used for DIG35 Metadata.
Cons	- Need to register a NEW private Tag ID.

I.2.1.7 Flashpix – Use Content Description Note

To record DIG35 Metadata in the Flashpix file format, it may be stored in the “Content description note” of “Content Description” property.

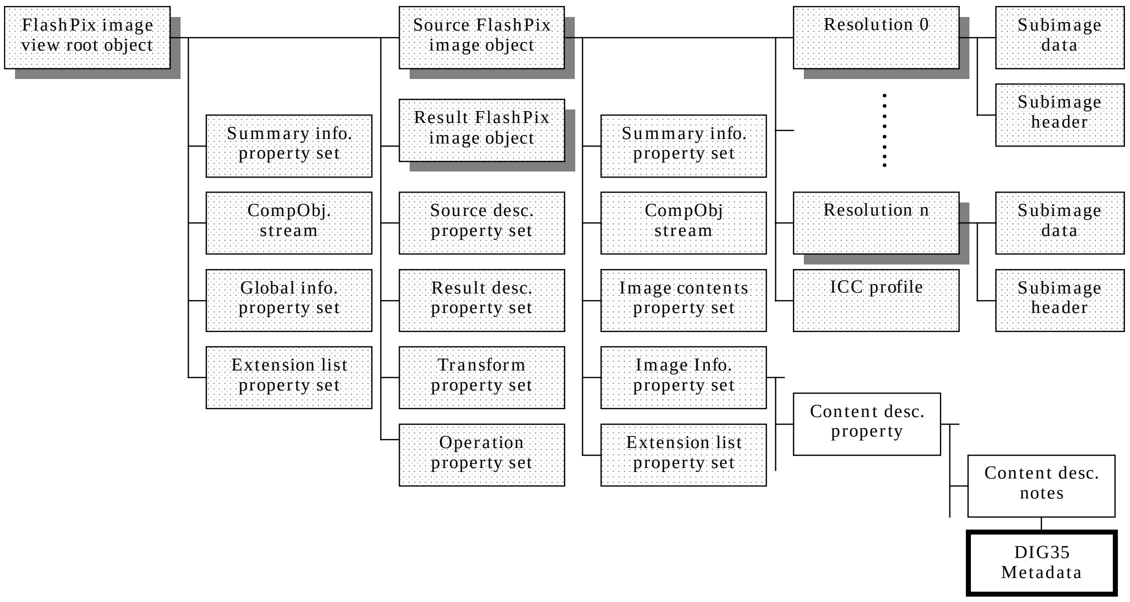

FlashPix image view root object
Source FlashPix image object
Resolution 0
Subimage data
Subimage header
Summary info. property set
Result FlashPix image object
Summary info. property set
CompObj. stream
Source desc. property set
CompObj stream
Resolution n
Subimage data
Global info. property set
Result desc. property set
Image contents property set
ICC profile
Subimage header
Extension list property set
Transform property set
Image Info. property set
Content desc. property
Operation property set
Extension list property set
Content desc. notes
DIG35 Metadata

Figure I-7: Flashpix file format with DIG35 metadata

Table I-8: Discussion items on Flashpix using Content Destription Note

Pros	- No need to add a NEW property set. - Exif’s “UserComment” can be convert to Flashpix’s “Description Note” property.
Cons	- Most Flashpix file format supporting applications do not read/write the property set.

I.2.1.8 Flashpix – Define New Property

DIG35 Metadata be recorded by defining a NEW property set. “DIG35 property set” for example, in the “Extension list”.

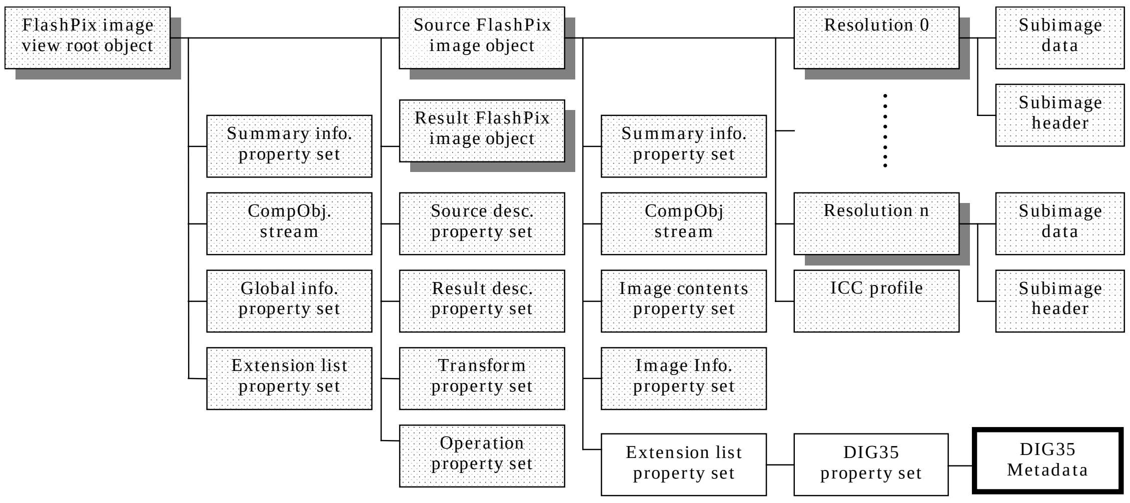

FlashPix image view root object
Source FlashPix image object
Resolution 0
Subimage data
Subimage header
Summary info. property set
Result FlashPix image object
Summary info. property set
CompObj. stream
Source desc. property set
CompObj stream
Resolution n
Subimage data
Global info. property set
Result desc. property set
Image contents property set
ICC profile
Subimage header
Extension list property set
Transform property set
Image Info. property set
Operation property set
Extension list property set
DIG35 property set
DIG35 Metadata

Figure I-8: Flashpix files format with DIG35 metadata

Table I-9: Discussion items on Flashpix defining a new property set

Pros	- Only DIG35 Metadata uses the new property set.
Cons	- Need to EXTEND the property set. - Most Flashpix file format supporting applications do not read/write the property set.

I.2.1.9 JPEG2000 – Use XML box

To record DIG35 Metadata in a JP2 file, it may be record in the "XML box".

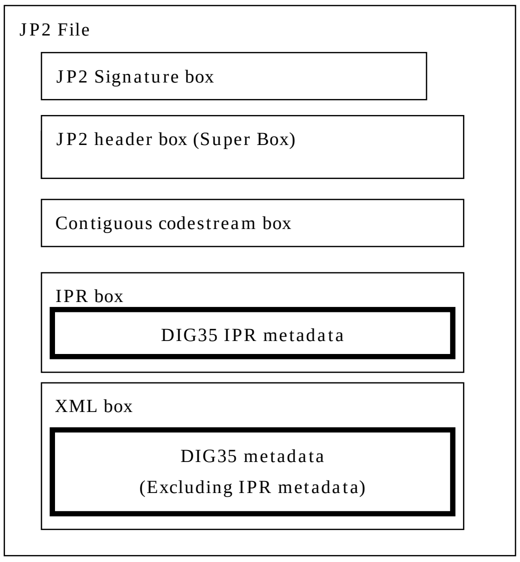


Figure I-9: JP2 file format with DIG35 metadata in XML box

Table I-10: Discussion items on using JP2 XML box

Pros	- JPEG2000 allows XML data (DIG35 Metadata) to be stored.
Cons	- May need to define XML Box / UUID boxes.

I.2.1.10 JPEG2000 – Define separate boxes for DIG35 sub-blocks

DIG35 may define box types for each sub-blocks in a JP2 file.

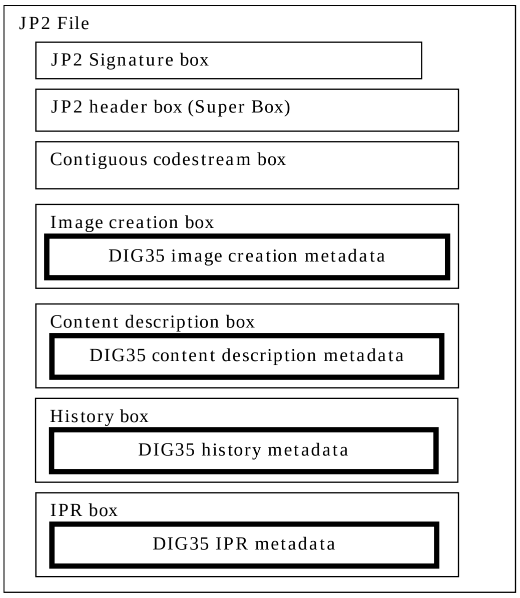


Figure I-10: JP2 file format with DIG35 metadata as separate boxes

Table I-11: Discussion items on defining separate DIG35 boxes for JP2

Pros	- DIG35 Metadata is identified by JPEG2000 box type
Cons	- Need to register box types to avoid box type collisions.

I.2.2 File Format Independent Association

I.2.2.1 File Format Independent

A file format independent implementation example. The metadata block consists of a header, the actual XML metadata and a footer. The footer is used to identify images that are associated with metadata and those that are not. The identification method consists of reading the fixed length footer, then scan the header where you can calculate from the footer and verify the check-sum, and finally verify the metadata body. By this process, the reader is able to verify if the metadata is correctly set and properly detach the metadata from the image file.

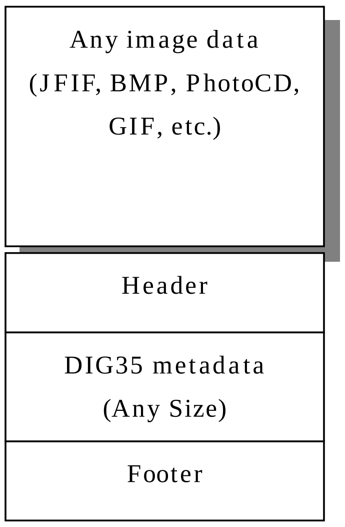


Figure I-11: A file format independent association method

Table I-12: Discussion items on File format independent association method

Pros	- Enable file format independent association - Can edit metadata without dealing with a image data. - Simple structure
Cons	- Becomes a different file as a whole

II Metadata Association

II.1 Introduction

This section is intended to illustrate many of the challenges attributed to associating metadata with images and the problems that are encountered relating to retention of image metadata throughout the lifecycle of that image.

The DIG35 has noted that there are three types of associations of metadata, internal, external and a combination internal and external (hybrid approach). All three types of associations are used in the industry today.

(1) **Internal**: Using the internal model, image metadata is stored with the image file itself. A good example is the storing of metadata in image file ‘headers’ or ‘tags’. Image metadata may also be attached to a file even if a pre-existing mapping of the image file data did not exist.

(2) **External**: The external model uses external references to associate metadata to an image file. Often times the image file name is used to link the external metadata and the image itself.

(3) **Hybrid**: Using both internal and external metadata for association.

As there are several challenges for each of the fore mentioned types of associations, each type will be discussed in detail below with examples of typical applications, challenges and technical issues for each.

II.2 Internal Model

Using the internal model, image metadata is stored in the same file as the image data itself. Images may be tagged or appended with metadata, in either case metadata is stored within the image file. The internal model is a robust solution for applications in which there is a known image workflow in a controlled environment. It is not ideal for uncontrolled environments in which different file formats, devices and applications are used

II.2.1 Typical Applications

As discussed above, the internal model is ideal for controlled environments where there may be only one file format, using a distinct set of devices and applications that are interoperable and retain the internal metadata throughout the image workflow. An example of this would be a business deployment of digital imaging hardware and applications, in which a single file format is used with a structured metadata template. This type of environment ensures that this metadata is retained throughout the image workflow.

The internal model is less than ideal for a consumer focused application in which a combination of capture devices, image editing application and services are used. In order to support each the centralized model in this environment, each device, application or service must be aware of the metadata and allow for the retention and versioning or this metadata.

II.2.2 Challenges and Technical Issues

II.2.2.1 File Format Dependent

File formats may not support an internal metadata structure or metadata structures may differ from file format to file format. One way to combat this problem, is to convert to a file format that supports an internal metadata structure (e.g. convert from JPEG to an Exif file for additional internal storage of metadata).

II.2.2.2 Metadata Aware Applications Required

For the internal model, applications, devices and services that work with a file that has internal metadata must be aware of this metadata and be able to retain, edit and version the metadata. Most applications in the industry today are metadata unaware, destroying or overwriting internal descriptive information when a file is saved. Further, software applications currently do not do a good job of retaining non-proprietary metadata and often times overwrite this metadata with their own proprietary application tags.

II.2.2.3 Metadata Versioning

Defining how old metadata is retained, edited or passed along in a given image workflow is a challenge.

II.2.2.4 Metadata Sharing & Collections

Complexities also occur when a single set of metadata must be assigned to multiple images or groups of images. To support internal metadata, each image must be tagged which is an inefficient process for a single set of metadata that can be associated with a group of images.

II.3 External Model

The external model provides a way to use external information along with a link to an image file in order to associate metadata with a digital image.

II.3.1 Typical Applications

Several common ways that the distributed model are used today include using a database structure to identify and associate metadata or using an external file (such as a text file, HTML file, XML file). It is the DIG35's goal to provide a common XML structure that can be used as a standard for distributed and centralized associations.

External models are ideal for using image files for one single application. For example, an online service that uses a database structure in order to store information about its users images is an example of using an external model.

External models are less than ideal for users that use their digital images in many applications as association files may get lost or broken when trying to tie a single image to multiple applications.

External models also fit well with images that are on read-only media such as CD-ROMs or DVDs. In these types of environments, images may stay on the read-only media, but the associated metadata must be assigned and managed else where in read-write form.

II.3.2 Challenges and Technical Issues

II.3.2.1 Broken Links

Using the distributed model, losing the link from one file to another or from a database to an image file is a common problem. This may occur if an image file is renamed and the corresponding metadata file or link is not updated to reflect this change. This is most likely to occur during the transmission of a digital image file (upload, download).

II.3.2.2 Different Association Types

As it is a challenge with the internal model to deal with the multiple file formats and their associated internal metadata structures, it is a challenge with the external model to deal with multiple association types (such as database, text file, etc.)

II.3.2.3 Metadata Versioning

Defining how old metadata is retained, edited or passed along in a given image workflow is a challenge. Also keeping link integrity is a key challenge when working with image metadata using the external model.

II.3.2.4 Different Types of Metadata

The external model poses an extra challenge when the metadata that is associated is not textual metadata. For example, if the metadata is an audio annotation then the association must either be done via a file naming convention or using another external file such as a text file.

II.4 Hybrid Model

The hybrid model uses both the internal and the external mechanisms for assignment of metadata.

II.4.1 Typical Applications

Typical applications include a workflow in which image submissions are coming from many disparate sources, some of which are currently using the internal model and others that are using the external model.

II.4.2 Challenges and Technical Issues

II.4.2.1 Applications Must Support Both Types

Using both types of structures provides its own set of challenges. Applications must now support two different types of associations, requiring complex checking and validation and some intelligence on which takes precedence when both types exist.

II.4.2.2 Metadata Versioning

As mentioned in both the internal and external models, metadata versioning is key issue.

II.5 File Format Dependence

The following tables illustrate the pros and cons of defining a standard that is either file format dependent or independent with relation to the models discussed above.

II.5.1 Internal model

	DEPENDENT	INDEPENDENT
PROS	- Legal image file format	- Single implementation applies to all formats. - Append - Need not worry about different formats having different metadata fields defined.
CONS	- Different file format may have different metadata schema. May not be able to pass one metadata set to another format. - Single API may be defined to attach DIG35 metadata, though need separate implementations for each file format to embed metadata	- Undefined format - May crash application - Image editing applications and services may need to be greatly redesigned.

II.5.2 External Model

	DEPENDENT	INDEPENDENT
PROS		- Support for all file formats regardless of internal structure.
CONS	- Limited file format support.	- Applications need to track file type or file extension

III Conformance

Different sets of conformance criteria exist for supporting the metadata defined in this specification:

- Conforming metadata definitions
- Conforming metadata implementations

III.1 Conforming Metadata Definitions

An image metadata definition is a *DIG35 conforming metadata definition* if it adheres to the specification described in this document and also has the capability of converting the metadata to the recommended implementation format described in this document.

Those who wish to conform at this level may choose a different implementation method other than XML.

III.2 Conforming Metadata Implementation

DIG35 metadata document is a *Conforming Metadata Implementation* if it adheres to the reference implementation and DTD specification described in this document and:

- is a well-formed XML document
- if all non-DIG35 namespace elements removed from the document, it is a valid XML document

III.2.1 Conforming DIG35 Stand-alone Files

It is a *Conforming DIG35 Stand-alone File* if:

- It is a conforming metadata implementation
- The root element is <METADATA>

III.2.2 Conforming DIG35 Included Files

A DIG35 metadata document that is included within another XML document is a *Conforming DIG35 Included File* if the DIG35 metadata document is properly extracted from the other XML document, conforms to the DIG35 DTD.

IV Other Encoding Methods

While DIG35 chose XML as a recommended implementation method especially for image metadata interchange, there are other means to exchange, store or stream image metadata within various systems. Several alternatives include:

- Tag based binary encoding (e.g. TIFF)
- Simple KEY-VALUE pair plain text
- Relational/Object database systems
- Other proprietary formats

V Future Plans

The following list consists of a list of technical issues not addressed by this specification though should be properly addressed in the future.

V-1. Audio as metadata, audio as part of image

V-2. Association scenarios when storage is non-writable

V-3. Compression of individual sub-blocks or the entire metadata

V-4. Authentication of metadata

V-5. Protection of metadata

VI Discussion Items

The DIG35 Initiative Group (IG) is presenting this document, DIG35 Specification Working Draft to several review communities: DIG members, JPEG2000 WG, and other image and technology industry members. Prior to finalizing the document in August of 2000 the DIG35 will incorporate feedback as accepted by the IG on the entire document. However, in developing the specification several issues were identified that had strong proponents for solutions in one or more manners. In review of this document DIG35 requests particular feedback on items noted below.

Keyword	**Short Description**	**Reference**
Importance	Scale of Importance/Relevance of the metadata element: is it needed and if so how to measure	Appendix VI.1
Redundant data	Redundant data between metadata and image data: which to take precedence	5.2.2
Validation	Metadata validation procedure: method to validate known metadata for an application	Appendix VI.2
GPS	GPS information requirements: are all GPS fields required? Which ones are useful/useless?	Annex A.2.3.6.5
Coordinate system	Normalizing the coordinate system: is this a requirement?	Annex A.2.3.8
Tag values	Consistency with tag values of other standards: for example, use the same values as JEIDA for capture source default options	None
TIFF/EP Tags	Several TIFF/EP Tags are not defined in this specification.	Appendix VI.3

VI.1 Importance/Relevance

DIG35 have discussed the introduction of an importance attribute for elements. We have had the following suggestions for the name of the field:

"Importance"

"Relevance"

This attribute would specify which of the metadata is important (or relevant to the image), and which is not. For example, an image with a tree in the background, and a house in the foreground, may contain the following keywords with associated importance/relevance:

House (importance = 0.9)

Tree (importance = 0.1)

If a picture of a house used by a real estate agent, then the address field would be marked as important. Whereas if an image of a house was taken as the background of "Craig standing next to his new car", then the address might still be listed, but would be listed as unimportant.

There are a number of ideas under discussion for the value stored in the importance field:

1. Use a range of values. If a value of zero is used for unimportant, and 1 for important, then what does 0.5 mean? Also - how can we specify a system where different people classifying an importance of 0.5 in multiple image files that the meaning is equivalent?
2. Use a rating within a file where the most important item was listed as a 1, and the next most important - 2, with the least import item being a large number. However, this system would still not allow comparing the importance of items in separate files. This system also falls down in that the user cannot specify that all of the content description is unimportant.

Some research will be done to determine how other metadata systems specify importance.

VI.2 Non-validated metadata

It is not possible for the creators of any file format to insist that everyone who opens a file understand all the metadata included in the file. In fact, it is often true that the creators of file formats go to great lengths to insure that different levels of readers exist or that file writers can put private data in the file that others may not understand. Often, the only requirement is that readers pass over metadata they don't understand.

This, while necessary, has led to significant problems. In some cases, metadata that is not understood will be dropped when a file is written back out. In other cases when the metadata is not dropped it will be inconsistent.

In an effort to provide a third alternative, metadata that is not understood could be marked as "non-validated". This could be done by marking each piece of metadata or moving the metadata to a non-validated tree. In either case, subsequent readers would be able to scan the non-validated data for information that might be useful, but it would be able to tell what data the latest reader couldn't update.

A simple example would be a clip operation (although this problem could be avoided by other means). If an image editing program couldn't read tags left by earlier editing programs, it could mark the earlier programs information that it couldn't understand as invalid. These non-validated tags could include content information. It could then clip the image and write it out. If the earlier editing program was to re-open the image it could recover the content information, but the user would have to somehow re-validate it---but they would not have to regenerate or re-enter the data.

VI.3 Undefined TIFF/EP Tags

The following TIFF/EP Tags are not defined in this specification.

- Orientation
- Self-timer Mode
- Battery Level
- Inter Color Profile

VII Syntax Notation

The XML syntax is defined in the following sections using an extended Backus-Naur Form (BNF). The BNF is used rather than using a DTD or Schema as it is the clearest method to define the restrictions on the metadata XML.

Note that where the BNF describes the exact details of the XML such as the '<' and '>', this is done to simply show what is required, and is not necessarily a restriction on the exact file syntax which must be used. For example, if an XML entity is used in the file, this still constitutes a valid metadata document.

The only extension to the BNF is the use of a BNF function. Where the following is used:

```
product_detail(N) ::= '<' N  language? timestamp '>'
                        MANUFACTURE?
                        MODEL?
                        SERIAL?
                      '</' N '>'
```

```
product_detail ('CAMERA_INFO')*
```

The expansion would replace `'CAMERA_INFO'` where the argument '*N*' is used within the `product_detail` function.

VIII Glossary

[Editor's note: To be completed later.]

IX Index

[Editor’s note: To be completed later.]

X DIG35 Working Group

This document has been prepared by the DIG35 Initiative Group (IG) of the Digital Imaging Group. The IG includes the following individuals (in alphabetical order):

Mo Asagari, PhotoChannel; Jean Barda, NETIMAGE; Craig Brown, Canon; Howard Bussey, Kodak; Sheldon Fisher, Microsoft; Scott Foshee, Adobe; Kats Ishii, Canon *(Chair and editor)*; Minoru Ishikawa, Fujifilm; Kiyoshi Kusama, Canon; Tim Long, Canon; George Lynch, HP; Kentaro Matsumoto, Canon; Robert Schultz, Digitella; David Wilkins, PictureIQ; Warren Whaley, Canon; Beverly Valdez, DIG

XI References

[1] Adobe Systems. *TIFF Revision 6.0 Final*. June 3, 1992

[2] ASTM. *Standard Practice for Electronic Interchange of Color and Appearance Data.* E1708-95.

[3] DIG. *Flashpix format specification.* Version 1.0.1. July 1997

[4] ICC. *ICC profile format specification.* ICC.1:1998–09

[5] ISO. *Codes for the representation of names of countries and their subdivisions – PART 1 Country Codes.* ISO 3166-1. 1 October 1997

[6] ISO. *Codes for the representation of names of countries and their subdivisions – PART 2 Country subdivision code.* ISO 3166-2. 15 December 1998

[7] ISO. *Codes for the representation of names of countries and their subdivisions – PART 3: Code for formerly used names of countries.* ISO 3166-3. 1 March 1999

[8] ISO. *Data elements and interchange formats – Information Interchange – Representation of dates and times*. ISO 8601. 1998

[9] ISO. *Photography – Electronic still picture cameras – Determination of ISO speed*. ISO 12232:1998.

[10] ISO. *Photography – Electronic still picture cameras – Resolution measurements*. ISO/DIS 12233.

[11] ISO. *Photography – Electronic still picture imaging – Removable memory – Part 2: Image data format – TIFF/EP*. ISO/DIS 12234-2

[12] ISO. *Photography – Electronic still picture cameras – Methods for measuring opto-electronic conversion functions (OECF's)*. ISO/DIS 14524.

[13] ISO. *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*. ISO/IEC 10918-1:1994.

[14] ISO. *Information technology – Digital compression and coding of continuous-tone still images: Registration of JPEG Profiles, SPIFF Profiles, SPIFF Tags, SPIFF colour Spaces, APPn Markers, SPIFF, Compression types and Registration authorities (REGAUT).* ISO/IEC 10918-4

[15] ISO. *Information technology -- JPEG2000 Image Coding System.* ISO/IEC CD 15444-1 :1999

[16] JEIDA. *Digital Still Camera File Format Standard (Exif).* Version 2.1. June 1998

[17] John S Denker. *See How It Files* 1996

[18] IETF. *Tags for Identification of Languages*. RFC 1766. March 1995

[19] IETF. *Uniform Resource Identifiers (URI).* RFC 2396. August 1998

[20] IETF. *UTF–8, A transformation format of ISO 10646.* RFC 2279. January 1998.

[21] IETF. *vCard MIME Directory Profile*. RFC 2426. September 1998

[22] WIPO. *Berne Convention for the Protection of Literary and Artistic Works.* Paris Act of July 24. 1971. amended September 28. 1979.

[23] WIPO. *World Intellectual Property Organization Copyright Treaty*. 1996.

[24] W3C. *Date and Time Formats*. <http://www.w3.org/TR/NOTE-datetime>. September 1997

[25] W3C. *Extensible Markup Language (XML 1.0).* Rec-xml-19980210

[26] W3C. *Namespaces in XML*. Rec-xml-names-19990114

[27] W3C. *XML Linking Language (XLink).* WD-xlink-20000221