OGC® DOCUMENT: 26-011

External identifier of this OGC® document: http://www.opengis.net/doc/PER/T21-GIMI-D203

Open Geospatial Consortium

OGC TESTBED-21 GIMI BASELINE SPECIFICATION REPORT

ENGINEERING REPORT

PUBLISHED

Version: 1.0
Submission Date: 2026-05-06
Approval Date: 2026-06-04
Publication Date: 2026-08-12
Editor: Joan Maso, Núria Julià

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

I ABSTRACT

This report defines the Baseline OGC Testbed-21 Specification for GIMI. The Report specifies requirements and encoding rules for using the High Efficiency Image File Format (HEIF) for the exchange of georeferenced or geocoded imagery accordingly with the current version of GIMI. The specification deviates from the Geographic High Efficiency Image Format (GeoHEIF) candidate specification that was proposed by the previous OGC Testbed-20 initiative. The core sections of this document specifies the GIMI specification using the OGC modular specification approach and define the necessary Requirements Classes. Additional annexes define the fundamental information necessary for reading HEIF files, the approaches for tiles and overviews in HEIF and the TAI related boxes.

Please note that this document is internally referred to as an OGC Testbed 21 Report or more simply “this Report” or “this candidate standard”.

II KEYWORDS

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, raster, GIMI, GeoHEIF, imagery, ISR, ISOBMFF, MPEG, HEIF, HEIC

PREFACE

The High Efficiency Image Format (HEIF) format, defined by Moving Picture Experts Group (MPEG), is a multipart data encoding that can support georeferenced imagery when used with the GeoHEIF extension that was proposed by the previous OGC Testbed-20 initiative. The GeoHEIF (Geographic High Efficiency Image Format) specification defines HEIF properties to include georeferencing information in one or more image items. These capabilities make it possible to include georeferencing information in a GIMI (GEOINT Imagery Media for ISR) imagery file, as GIMI is a profile of the HEIF standard. The OGC Testbed-21 initiative builds upon the achievements of the previous Testbed to define a baseline GIMI specification for the initiative that can then inform future versions of GIMI. This report documents the baseline specification of GIMI for this Testbed.

IV SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Universitat Autònoma de Barcelona (CREAF)

V SUBMITTERS

All questions regarding this submission should be directed to the editor or the submitters:

NAME	AFFILIATION	OGC MEMBER
Núria Julià Selvas	Universitat Autònoma de Barcelona (CREAF)	Yes
Joan Masó Pau	Universitat Autònoma de Barcelona (CREAF)	Yes

VI SECURITY CONSIDERATIONS

No security considerations are made in this Report. Future work could consider including information about security labeling, digital rights management (DRM), or methods for encrypting images.

1

SCOPE

1 SCOPE

This report defines the Baseline OGC Testbed-21 Specification for GIMI when the Testbed 21 started. GIMI extends HEIF by adding International Atomic Time (TAI) nanosecond formatted timestamps, information security markings of a dataset (including codewords, file control and handling, releasing instructions, declassification, and downgrading parameters), and content identification (Content ID). As HEIF has many possible ways of combining information and allows for several encodings, GIMI limits the content increasing the interoperability of the implementation. The scope of this document is limited to HEIF files containing still imagery, image sequences and motions imagery.

Advancements made during the Testbed 21 initiative are out of scope of this document, but are documented in a separate Testbed-21 report (Please refer to “OGC 26-011 Advancement of the GIMI Standard Report”). Convergency of the Geographic High Efficiency Image Format (GeoHEIF) candidate specification proposed by the previous OGC Testbed-20 initiative is also out of scope of this document.

2 CONFORMANCE

2 CONFORMANCE

This report defines several conformance classes for the GIMI format.

Requirements for one standardization target types (GIMI File) are considered:

- https://www.opengis.net/spec/GIMI/1.0/req/gimi: GIMI core
- https://www.opengis.net/spec/GIMI/1.0/req/content-id: global content ID system
- https://www.opengis.net/spec/GIMI/1.0/req/tai: international atomic time (TAI)
- https://www.opengis.net/spec/GIMI/1.0/req/still-imagery: Still imagery
- https://www.opengis.net/spec/GIMI/1.0/req/imagery-sequences: Image Sequences
- https://www.opengis.net/spec/GIMI/1.0/req/motion-imagery: General Motion Imagery
- https://www.opengis.net/spec/GIMI/1.0/req/class0-motion-imagery: Class 0 Motion Imagery
- https://www.opengis.net/spec/GIMI/1.0/req/class1-motion-imagery: Class 1 Motion Imagery
- https://www.opengis.net/spec/GIMI/1.0/req/class2-motion-imagery: Class 2 Motion Imagery

Conformance with this candidate standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

In order to conform to this candidate standard, a software implementation shall choose to implement:

- Any one of the conformance levels specified in Annex A (normative).

All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

3 NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

International Organization for Standardization (committee): ISO/IEC 14496-12:2022, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2022). https://www.iso.org/standard/83102.html.

International Organization for Standardization (committee): ISO/IEC 23008-12:2022, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 12: Image File Format*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2022). https://www.iso.org/standard/83650.html.

GEOINT Imagery Media for Intelligence, Surveillance, and Reconnaissance (ISR) (GIMI) — Volume 1 — Profile of ISOBMFF & HEIF — (2024-10-23) — Version 1.0.0 — NATIONAL CENTER FOR GEOSPATIAL INTELLIGENCE STANDARDS

4

TERMS, DEFINITIONS AND ABBREVIATED TERMS

4 TERMS, DEFINITIONS AND ABBREVIATED TERMS

This document uses the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

4.1. **4CC**

ASCII encoded “four-character code” indicating data formatting requirements. Applications implement the 4CC code as a 32-bit big-endian unsigned integer with the first character in the MSB, last character in the LSB. There is no string termination

(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.2. **box**

object-oriented container with a unique type identifier and length.

(source: MPEG)

4.3. **brand**

an identifier in the FileTypeBox of a HEIF file that identify a specification to which the file complies. A HEIF file can have one brand and several sub-brands.

4.4. codec

hardware or software component providing both an encode and decode function for imagery, video or other media types.

NOTE; Shorthand for coder-decoder

(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.5. content id

universally unique identifier providing enterprise-wide search and discovery ability. Content IDs apply to a specific sequence of bits representing a piece of content.

(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.6. coordinate reference system

coordinate system that is related to an object by a datum

(source: OGC18-005r8,clause 3.1)

Note 1 to entry: Geodetic and vertical datums are referred to as reference frames.

Note 2 to entry: For geodetic and vertical reference frames, the object will be the Earth. In planetary applications, geodetic and vertical reference frames may be applied to other celestial bodies

4.7. image sequence

ordered series of coded images which may be associated with advisory timing and in which images may use inter prediction.

(source: MPEG)

4.8. international atomic time

high-precision, absolute, time scale derived from hundreds of precise atomic clocks from around the world and maintained as closely as possible to the Système International (SI) second. Current practice achieves a maximum deviation of approximately one second every 100 million years.

(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.9. item

data which does not require timed processing, as opposed to sample data. Items may be an image or an instance of metadata

(source: MPEG)

4.10. lossless compression

form of compression where decompression results in a bit-wise identical result to the original.
(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.11. lossy compression

form of image compression where decompression results in an image containing differences, to some degree, from the original.

(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.12. motion Imagery

sequence of images, that when viewed (e.g., with a media player) must have the potential for providing informational or intelligence value. This implies the images composing the Motion Imagery are: (1) from sensed data, and (2) relate to each other both in time and in space. The sequence of images enables detecting motion of the sensor and/or objects within.

(source: MISB)

4.13. resource description framework

semantic web data format standard for representing interconnected data.

(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.14. still image

two or more-dimensional rectangular array of pixels derived from sensed phenomena indexed by row and column.

(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.15. still imagery

category of single images lacking the continuous temporal collection aspect of Motion Imagery. Without additional images, determining motion in a scene is much more difficult.

(source: NGA.STND.0076-01_V1.0.0_GIMI)

4.16. sidecar

file containing only metadata (e.g. the georeference information) and pertaining to one or more GIMI files.

(source: NGA.STND.0076-01_V1.0.0_GIMI)

Note 1 to entry: the same content can also be embedded in the 'mdat' box of the HEIF file

4.17. **tile**

geometric shape with known properties that may or may not be the result of a tiling (tessellation) process. A tile consists of a single connected “piece” (topological disc) without “holes” or “lines”.

(source: OGC 19-014r3,clause 4.13)

Note 1 to entry: In the above definition, the term hole means that any given tile in a tileset cannot have sub-tiles or exclusion areas. The use of the term `hole’ in the above definition should not be confused with exclusion areas in a polygon geometry (aka donuts or islands).

Note 2 to entry: For the purpose of this standard, tile in a tileset do not overlap

4.18. Abbreviated terms

2D	Two dimensions or two-dimensional
3D	Three dimensions or three-dimensional
4CC	Four Character Code
AVIF	AV1 Image File Format
CIS	Coverage Implementation Schema
CRS	Coordinate Reference System
CSCRNA	Corner Footprint
EPSG	European Petroleum Survey Group
GeoHEIF	Geographic High Efficiency Image Format
GEOINT	Geospatial-Intelligence
GIMI	GEOINT Imagery Media for ISR
HEIF	High Efficiency Image File Format
HEVC	High Efficiency Video Coding
ISO	International Organization for Standardization
ISOBMFF	ISO Base Media File Format

ISR	Intelligence, Surveillance, and Reconnaissance
JPEG	Joint Photographic Experts Group
MPEG	Moving Picture Experts Group
RDF	Resource Description Framework
TAI	International Atomic Time
TC	Technical Committee
URI	Uniform Resource Identifier

4.19. Identifiers

The normative provisions in this standard are denoted by the URI

`https://www.opengis.net/spec/GIMI/1.0-draft2`

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

5 OVERVIEW

5 OVERVIEW

The flexibility defined in the HEIF format determines the interoperability of the applications trying to read it. To improve the interoperability of HEIF files for geospatial applications, the GEOINT Imagery Media for ISR (GIMI) Profile of ISOBMFF & HEIF was defined by the National Geospatial Intelligence Agency (NGA) and adopted by the Geospatial-Intelligence (GEOINT) Standards Working Group (GWG). The GWG serves as the advisory body for advancing and enabling GEOINT standards across the US National System for Geospatial-Intelligence (NSG).

The GIMI standard proposes the mandatory inclusion of International Atomic Time (TAI) nanosecond formatted timestamps, information security markings of a dataset (including codewords, file control and handling, releasing instructions, declassification, and downgrading parameters), and content identification (Content ID) information supporting enterprise-wide uniqueness, traceability, effective search/discovery, and file releasability.

Requirements for declaring and encoding of metadata, along with requirements restricting the types and usage of coder-decoders (codecs) per type of media content ensure that files conformant to this standard maximize interoperability.

To facilitate interoperability with GEOINT Enterprise ontology structured metadata systems, this standard defines methods for encoding and carriage of metadata internal to the HEIF file, as well as methods for referencing metadata carried external to the HEIF file (i.e., a “sidecar” file in of Resource Description Framework (RDF) content encoded in the “TriG syntax”, and extension of “turtle”). GIMI defines essential metadata (content identification and high precision timing), security metadata, and generalized application metadata.

5.1. Description of the two ways to encode georeference in HEIF and have a GeoHEIF file

This document describes the need to have requirements classes for the georeferencing imagery. In Testbed 20, a method based in the use of binary boxes that are ItemPropertyTypes was suggested. However, in Testbed 21 a second method, based on an RDF document was introduced. This baseline includes a simple way to encode the georeference as RDF. During the Testbed 21 initiative, the way that RDF documents should be encoded has considerably evolved and will be documented in the Testbed 21 — Advancement of the GIMI Standard Report.

6

REQUIREMENT CLASSES FOR GIMI CORE

6 REQUIREMENT CLASSES FOR GIMI CORE

6.1. Underlying GIMI core Requirement

This section addresses the core HEIF Requirements Class applicable to all GeoHEIF instances (i.e., files).

A GIMI file is a HEIF file and inherits the file structure as described in the corresponding portion of the ISOBMFF and HEIF specifications. In addition, a brand is added to the file to specify conformance to the requirements specified in this Report.

REQUIREMENTS CLASS 1: REQUIREMENTS CLASS 'CORE'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/core`
TARGET TYPE	GIMI File
PREREQUISITES	ISO14496-12,ISO/IEC 14496-12:2022, Information technology — Coding of audio-visual objects — Part 12: ISO base media file format ISO/IEC 23008-12:2022, Information technology — High efficiency coding and media delivery in heterogenous environments — Part 12: Image File Format ISO/IEC 23001-17, Information technology — MPEG Systems technologies — Part17: Carriage of uncompressed video and images in ISO Base Media File Format
NORMATIVE STATEMENTS	Requirement 1: `/req/core/follow-ISOBMFF` Requirement 2: `/req/core/follow-HEIF` Requirement 3: `/req/core/geo1-brand`

6.1.1. File format

The following two requirements specify the file format used.

REQUIREMENT 1	
IDENTIFIER	`/req/core/follow-ISOBMFF`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/GIMI/1.0/req/core`

REQUIREMENT 1	
A	A GIMI file SHALL be compliant with the ISO Base Media File Format (ISOBMFF) specification.

REQUIREMENT 2	
IDENTIFIER	`/req/core/follow-HEIF`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/GIMI/1.0/req/core`
A	A GIMI file SHALL be compliant with the High Efficiency Image File Format (HEIF) specification.

6.1.2. The `geo1` brand

ISOBMFF files use brands in the FileTypeBox (`ftyp`) to signal interoperability requirements. The FileTypeBox contains a `major_brand` and a list of `compatible_brands`. Each brand is an unsigned 32 bit integer value that is interpreted as a four character code (4CC), similar to the way boxes are identified, but in a different implicit namespace.

NOTE: Under some circumstances, brands can occur in other boxes for similar use, such as the BrandProperty (`brnd`). The requirements in this section apply to use of those brands in context.

REQUIREMENT 3	
IDENTIFIER	`/req/core/geo1-brand`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/GIMI/1.0/req/core`
A	A GIMI file SHALL include the `geo1` brand in the FileTypeBox (`ftyp`) in the compatible brands array.

The `geo1` brand indicates a file is conformant to version 1 of the standard NGA.STND.0076_1.0:2024. A GIMI file may contain still imagery and motion imagery. For still imagery, the scope of the ´geo1´ brand includes rectangular, uncompressed, and compressed forms, including monochrome, color visible, panchromatic, MSI, HSI, and SAR. For motion imagery, the ´geo1´ brand includes rectangular forms of Class 0 (uncompressed), Class 1 (broadcast, compressed), and Class 2 (SEMI, compressed).

7

REQUIREMENT CLASSES FOR CONTENT IDENTIFICATION

7 REQUIREMENT CLASSES FOR CONTENT IDENTIFICATION

GIMI-essential metadata establishes an identity foundation for every item of media content. GIMI-essential metadata provides two capabilities: universal identification of media content and timestamp labeling of media content. Universal identification requires assigning a Content Identifier (Content ID) to every media data element. ISOBMFF defines an internal entity ID system to link media together. However, the ID system does not support global uniqueness. The purpose of this requirements class is to define global Content IDs.

REQUIREMENTS CLASS 2: REQUIREMENTS CLASS 'CONTENT IDENTIFICATION'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/content-id`
TARGET TYPE	GIMI File
PREREQUISITE	https://www.opengis.net/spec/GIMI/1.0/req/gimi
NORMATIVE STATEMENTS	Requirement 4: `/req/content-id/unif-brand` Requirement 5: `/req/content-id/for-each-item` Requirement 6: `/req/content-id/content`

7.1. The unif brand

The `unif` brand indicates the unified implementation and handling of IDs across file-scoped MetaBox items, tracks, track groups, and entity groups.

REQUIREMENT 4	
IDENTIFIER	`/req/content-id/unif-brand`
INCLUDED IN	Requirements class 2: `https://www.opengis.net/spec/GIMI/1.0/req/content-id`
A	A GIMI file SHALL include the `unif` brand in the FileTypeBox (`ftyp`) in the compatible brands array.

The Content IDs provide a means to identify and locate content, attach security markings to content, and link to related ontology structured data, including when contained in external/ sidecar files.

A Content ID identifies a specific sequence of bits and should not be changed if the piece of content they represent does not change. If the bits change in any way, including the removal of bits or the addition of more bits, systems must generate a new Content ID for the modified content.

REQUIREMENT 5	
IDENTIFIER	`/req/content-id/for-each-item`
INCLUDED IN	Requirements class 2: `https://www.opengis.net/spec/GIMI/1.0/req/content-id`
A	A Content identifier (ContentID) SHALL be assigned to every declared item having a mediatype (image, metadata, and region, track, track portion, track sample, item component, and sample component).
B	ContentIDs SHALL be stored for each unique piece of content using built-in capabilities of ISOBMFF and HEIF as specified in the table Table 1.
C	Depending on the ContentID type, the methods of declaring ContentIDs detailed in the table Table 2 SHALL be used.

Table 1 — Declaration methods for ContentIDs

Media Type	Content ID Type	Declaration Method
Items: still images, single instance metadata, and still image regions	ItemContentID	`uuid` extended type item property
Tracks: video, image sequences, timed metadata, and audio	TrackContentID	URI-defined metadata item in the track's MetaBox
Track Portions: video, image sequences, timed metadata, and audio	TrackPortionContentID	URI-defined metadata item in the track's MetaBox
Samples: images and metadata	SampleContentID	Sample auxiliary information
Image Item Components: still images, 2D metadata arrays	ItemComponentContentID	`uuid` extended type item property
Track Imagery Components: video, image sequences, 2D metadata array sequences	TrackComponentContentID	URI-defined metadata item in the track's MetaBox

Table 2 — ContentID types and their declaration parameters

Content ID	Declaration Parameter	Form

ItemContentID	0×261ef3741d975bbaacbd9d2c8ea73522	uint(8)[16]
TrackContentID	‘urn:uuid:15beb8e4-944d-5fc6-a3dd-cb5a7e655c73’	‘urn:uuid:<uuid>’
TrackPortionContentID	‘urn:uuid:7e26ca23-c8f9-5e55-8776-cd342f81e16d’	‘urn:uuid:<uuid>’
SampleContentID	‘suid’	4CC
ItemComponentContentID	‘0×9db9dd6e373c5a4e811021fc83a911fd’	uint(8)[16]
TrackComponentContentID	‘urn:uuid:fef58f02-43a6-5aaf-a891-099b1953d1f6’	‘urn:uuid:<uuid>’

The value of the ContentID can have a URN UUID conformant to ISO/IEC 9834-8 and NGA.RP.0001.

REQUIREMENT 6	
IDENTIFIER	`/req/content-id/content`
INCLUDED IN	Requirements class 2: `https://www.opengis.net/spec/GIMI/1.0/req/content-id`
A	The ContentID SHALL be a string formatted canonical form URN UUID (’urn:uuid:<uuid>’) conformant to ISO/IEC 9834-8 and NGA.RP.0001.

8

REQUIREMENT CLASSES FOR INTERNATIONAL ATOMIC TIME

8 REQUIREMENT CLASSES FOR INTERNATIONAL ATOMIC TIME

GIMI-essential metadata establishes an identity foundation for every item of media content. GIMI-essential metadata provides two capabilities: universal identification of media content and timestamp labeling of media content. Timestamp labeling requires assigning an absolute timestamp, i.e., an International Atomic Time timestamp (TAI timestamp), to every media content data element.

Within ISOBMFF, multiple forms of timing information exist within a file. ISOBMFF defines internal mechanisms for labeling absolute time, based on UTC, and for creating timelines for timed media, which supports the playout of the media. However, ISOBMFF timeline is not designed to facilitate high precision, nanosecond sensor measurements, analysis, and exploitation. With reliance on 32-bit integers, the longest presentation possible using nanosecond timing is less than five seconds. By implementing TAI timestamps independent of the ISOBMFF built-in timeline, GIMI achieves the necessary high precision measurement timing as well as being able to leverage the existing timeline tools for playout with standard consumer commercial video players.

The TAI time system is synchronous with the GPS time system (with a fixed offset of 19 seconds) and avoids problems associated with leap seconds, which occasionally occur in the UTC time system. Software applications with requirements for UTC time, such as for human display, convert TAI timestamps to UTC time with the proper leap second corrections and adjustments for time zones, etc. Applications need to be mindful of properly conveying to users the type and source of time when displayed.

TAI_timestamp is a 64-bit unsigned integer representing the number of nanoseconds since the TAI epoch of 1958-01-01T00:00:00.0. Each nanosecond is one billionth of a Standard International (SI) Second.

REQUIREMENTS CLASS 3: REQUIREMENTS CLASS 'INTERNATIONAL ATOMIC TIME'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/tai`
TARGET TYPE	GIMI File
PREREQUISITE	https://www.opengis.net/spec/GIMI/1.0/req/gimi
NORMATIVE STATEMENTS	Requirement 7: `/req/tai/for-each-item` Requirement 8: `/req/tai/content`

REQUIREMENT 7	
IDENTIFIER	`/req/tai/for-each-item`
INCLUDED IN	Requirements class 3: `https://www.opengis.net/spec/GIMI/1.0/req/tai`
A	A TAI SHALL be assigned to every declared motion imagery sample, image sequence sample, timed metadata items.
B	The assigned TAI Timestamps SHALL we encoded in a TAITimestampPacket descriptive item property as sample auxiliary information.
C	Each TAI timestamp SHALL be associated to a clock details encoded in a TAIClockInfoBox.
D	Each TAI timestamp SHALL indicate the beginning of a physical measurement (e.g. the start of exposure for an imaging sensor)
E	The items' TAITimestampPacket and its TAIClockInfoBox SHALL persist.
F	The timestamp_is_modified bit in the TAITimestampPacket SHALL be set to 1 when a timestamp previously written into a file is changed.

REQUIREMENT 8	
IDENTIFIER	`/req/tai/content`
INCLUDED IN	Requirements class 3: `https://www.opengis.net/spec/GIMI/1.0/req/tai`
A	High precision timing information SHALL consist in a media timestamp and clock details conformant to ISO/IEC 23001-17 Amd1 Annex D
B	TAI_timestamp SHALL be expressed as a 64-bit unsigned integer representing the number of nanoseconds since the TAI epoch of 1958-01-01T00:00:00.0. Each nanosecond is one billionth of a Standard International (SI) Second.
C	The clock details include the clock pedigree and per timestamp status metadata

The data structures for TAIs are detailed in an annex of this document.

9

REQUIREMENT CLASSES FOR STILL IMAGERY

9 REQUIREMENT CLASSES FOR STILL IMAGERY

This section covers the nature of still imagery in general that can be in the forms for uncompressed and compressed imagery when stored in a HEIF file.

REQUIREMENTS CLASS 4: REQUIREMENTS CLASS 'STILL IMAGERY'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/still-imagery`
TARGET TYPE	GIMI File
PREREQUISITES	Requirements class 2: `https://www.opengis.net/spec/GIMI/1.0/req/content-id` Requirements class 3: `https://www.opengis.net/spec/GIMI/1.0/req/tai`
NORMATIVE STATEMENTS	Requirement 9: `/req/still-imagery/mif2-brand` Requirement 10: `/req/still-imagery/codecs`

9.1. The mif2 brand

The brand is included in the FileTypeBox (the first box that a HEIF reader should read) and provides an indication of the content of the HEIF file. In addition to the main brand, the same box provides additional compatible brands.

The `mif2` brand indicates the presence of still imagery in the GIMI File.

REQUIREMENT 9	
IDENTIFIER	`/req/still-imagery/mif2-brand`
INCLUDED IN	Requirements class 4: `https://www.opengis.net/spec/GIMI/1.0/req/still-imagery`
A	A GIMI file including still imagery SHALL include the `mif2` brand in the FileTypeBox (`ftyp`) in the compatible brands array.

The 'mif2' brand represents interoperability requirements for image items and associated metadata items. 'mif2' represents a baseline for Still Imagery support in this candidate standard. The HEIF standard documents the specifics of the branding differences.

9.2. Codecs supported for Still Imagery

ISOBMFF and HEIF standards are agnostic to image encoding methods and support a wide variety of existing codecs. This candidate standard allows a selected sub-set of codecs to balance functional coverage, supporting legacy needs, and maximizing interoperability.

Table 3 — List of codecs approved for use with Still Imagery

Encoding Method	Codec Standard	Carriage in GIMI
Uncompressed Video and Images in ISOBMFF	ISO/IEC 23001-17	ISO/IEC 23001-17
JPEG 2000 (Part 1 & Part 2)	ISO/IEC 15444-1, ISO/IEC 15444-2	ISO/IEC 15444-16
High Throughput JPEG 2000	ISO/IEC 15444-15	ISO/IEC 15444-16
HEVC	ISO/IEC 23008-2	ISO/IEC 23008-12
AVC	ISO/IEC 14496-10	ISO/IEC 23008-12

REQUIREMENT 10	
IDENTIFIER	`/req/still-imagery/codecs`
INCLUDED IN	Requirements class 4: `https://www.opengis.net/spec/GIMI/1.0/req/still-imagery`
A	Where an GIMI file contains still imagery, the coding of the still imagery SHALL conform to one of the codec standards listed Table 3.

This candidate standard addresses a broad range of imagery forms and types including common one and three band imagery, multi and hyperspectral imagery, SAR imagery, imagery with unusually high resolutions and/or high dynamic range, and imagery with extended definition storage formats, such as floating point and complex values.

The ISO/IEC 23001-17 uncompressed codec, as well as numerically lossless implementations of other codecs, such as JPEG 2000, are for applications where preservation of content and quality of imagery is of primary concern. For situations where storage and transmission efficiency are of primary consideration, lossy compression options, including those with visually lossless compression settings, are available.

JPEG 2000 (including Part 1, Part 2, and HTJ2K) is a primary set of codec capabilities for still imagery applications. The underlying wavelet technology supports high dynamic range, high

density images, high numbers of components, and unique methods for accessing regions-of-interest without having to load and decompress the entire image. To support the need for greater processing efficiency, this candidate standard supports the use of High-Throughput JPEG 2000 (HTJ2K).

While HEVC and AVC are principally video codecs, they do support still image (and image sequence) compression. These codecs find use in applications requiring high interoperability, such as disaster relief, as well as applications for extracting and storing a single frame from an HEVC or AVC coded image.

In cases of an uncommon form of pixel (high bit depth, floating point, etc.) or codec profile (visually-eccentric imagery), it may be advantageous to re-encode the image using a common codec, such as HEVC, and include this visually-normal imagery as an alternate version of an image item or track. This improves interoperability with commodity devices and software allowing a higher percentage of applications to view some version of imagery in the file. The utility of this approach is dependent on a variety of issues related to workflow, security, stakeholder resources, long term storage cost, etc. In applications requiring the highest level of fidelity, such as exploitation, developers need to take care to ensure users are working with the highest quality image available.

RECOMMENDATION 1	
IDENTIFIER	`/rec/still-imagery/alternative-codec`
A	In cases of an uncommon form of pixel (high bit depth, floating point, etc.) or codec profile (visually-eccentric imagery), the GIMI format SHOULD re-encode the image using a common codec, such as HEVC, and include this visually-normal imagery as an alternate version of an image item.
B	The visually-normal version SHOULD be associated to the full version by adding an entity group of type alternate ('altr') in a groups list box ('grpl'), containing both forms of the image item to inform the visually-normal image item is an alternate, more interoperable, and displayable form of the visually-eccentric image item.

10

REQUIREMENT CLASSES FOR IMAGE SEQUENCES

10 REQUIREMENT CLASSES FOR IMAGE SEQUENCES

An image sequence is an ordered series of images. The following traits support image sequences:

1. Sequences are an ordered series of coded images;
2. Sequences may be associated with advisory timing; and
3. The coding of images may use inter-prediction.

REQUIREMENTS CLASS 5: REQUIREMENTS CLASS 'IMAGERY SEQUENCES'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/imagery-sequences`
TARGET TYPE	GIMI File
PREREQUISITES	Requirements class 2: `https://www.opengis.net/spec/GIMI/1.0/req/content-id` Requirements class 3: `https://www.opengis.net/spec/GIMI/1.0/req/tai`
NORMATIVE STATEMENTS	Requirement 11: `/req/imagery-sequences/msf1-brand` Requirement 12: `/req/imagery-sequences/codecs`

10.1. The msf1 brand

The 'msf1' brand indicates the presence of imagery sequences in the GIMI file.

REQUIREMENT 11	
IDENTIFIER	`/req/imagery-sequences/msf1-brand`
INCLUDED IN	Requirements class 5: `https://www.opengis.net/spec/GIMI/1.0/req/imagery-sequences`
A	A GIMI file including imagery sequences SHALL include the `msf1` brand in the FileTypeBox (`ftyp`) in the compatible brands array.

10.2. Codecs supported for Imagery Sequences

ISOBMFF and HEIF standards are agnostic to image encoding methods and support a wide variety of existing codecs. This candidate standards allows a select sub-set of codecs to balance the trade space of functional coverage, supporting legacy needs, and maximizing interoperability.

REQUIREMENT 12	
IDENTIFIER	`/req/imagery-sequences/codecs`
INCLUDED IN	Requirements class 5: `https://www.opengis.net/spec/GIMI/1.0/req/imagery-sequences`
A	Where a GIMI file contains imagery sequences, the coding of the imagery sequences SHALL conform to one of the codec standards listed Table 4.

Table 4 — List of codecs approved for use with Imagery Sequences

Encoding Method	Codec Standard	Carriage in GIMI
Uncompressed Video and Images in ISOBMFF	ISO/IEC 23001-17	ISO/IEC 23001-17
JPEG 2000 (Part 1 & Part 2)	ISO/IEC 15444-1, ISO/IEC 15444-2	ISO/IEC 15444-16
High Throughput JPEG 2000	ISO/IEC 15444-15	ISO/IEC 15444-16
HEVC	ISO/IEC 23008-2	ISO/IEC 23008-12
AVC	ISO/IEC 14496-10	ISO/IEC 23008-12

11

REQUIREMENT CLASSES FOR MOTION IMAGERY

11 REQUIREMENT CLASSES FOR MOTION IMAGERY

Motion Imagery is categorized in the Motion Imagery Standards Profile (MISP) into defined types or classes.

- **Class 0 Motion Imagery (MISP)**: represents uncompressed forms of Motion Imagery, metadata, audio, and suitable containers. It allows for unconstrained image characteristics such as unrestricted pixel value range, Number of bands, number of pixels per image, and number of images per second. Examples include 14-bit infrared, Bayer, 3-Band High Definition.
- **Class 1 Motion Imagery (MISP)**: broadcast forms of compressed Motion Imagery, metadata, audio, and suitable containers, such as SD, HD, 4K, etc. It is applicable when delivering monochrome and color Motion Imagery in cases where the transmission bandwidth prohibits the use of Class 0 Motion Imagery. It allows for constrained image characteristics such as limits in pixel value range, number of bands, and number of images per second. Example is H.264/AVC compressed airborne Motion Imagery with three bands of color.
- **Class 2 Motion Imagery (MISP)**: compressed Motion Imagery, metadata, audio, and suitable containers, with unconstrained image characteristics. Unlike Class 1 Motion Imagery, Class 2 Motion Imagery allows for a non-constrained set of image characteristics. Examples include high frame rate scientific imaging, Large Volume Motion Imagery, and high bit-depth compressed infrared.
- **Class 3 Motion Imagery (non-MISP)**: forms of compressed Motion Imagery not conformant to the MISP, represents sources such as cell phones, mobile devices, security or surveillance cameras which may use formats, compression, containers, and other technologies that do not conform to the MISP.

11.1. General Requirement Class for Motion Imagery

REQUIREMENTS CLASS 6: REQUIREMENTS CLASS 'MOTION IMAGERY'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/motion-imagery`
TARGET TYPE	GIMI File

REQUIREMENTS CLASS 6: REQUIREMENTS CLASS 'MOTION IMAGERY'	
PREREQUISITES	Requirements class 2: `https://www.opengis.net/spec/GIMI/1.0/req/content-id` Requirements class 3: `https://www.opengis.net/spec/GIMI/1.0/req/tai`
NORMATIVE STATEMENT	Requirement 13: `/req/motion-imagery/isof-brand`

The 'isob' brand indicates the presence of the Media Tracks, including motion imagery in the GIMI file. A file that carries motion imagery has to include a MovieBox and a TrackBox. To signal a track is carrying Motion Imagery, the handler type for the track is set to 'vide'.

REQUIREMENT 13	
IDENTIFIER	`/req/motion-imagery/isof-brand`
INCLUDED IN	Requirements class 6: `https://www.opengis.net/spec/GIMI/1.0/req/motion-imagery`
A	A GIMI file including media track content SHALL include the `isof` brand in the FileTypeBox (`ftyp`) in the compatible brands array.

The 'isob' brand addresses boxes defined in ISOBMFF, many of which directly address the carriage of video and other media tracks.

11.2. Class 0 Motion Imagery

The Class 0 Motion Imagery (MISP) represents uncompressed forms of Motion Imagery, metadata, audio, and suitable containers. It allows for unconstrained image characteristics such as unrestricted pixel value range, Number of bands, number of pixels per image, and number of images per second. Examples include 14-bit infrared, Bayer, 3-Band High Definition.

REQUIREMENTS CLASS 7: REQUIREMENTS CLASS 'CLASS 0 MOTION IMAGERY'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/class0-motion-imagery`
TARGET TYPE	GIMI File
PREREQUISITE	/req/motion-imagery
NORMATIVE STATEMENT	Requirement 7-1: `/req/motion-imagery/codecs-class0`

Class 0 Motion Imagery applications use the Uncompressed Video and Images in ISOBMFF (ISO/IEC 23001-17) codec.

REQUIREMENT 14	
IDENTIFIER	`/req/motion-imagery/codecs_class0`
A	Where a GIMI file contains motion imagery class 0, the coding of the motion imagery SHALL conform to one of the codec standards listed Table 5.

Table 5 — List of codecs approved for use with Class 0 Motion Imagery

Encoding Method	Codec Standard	Carriage in GIMI
Uncompressed Video and Images in ISOBMFF	ISO/IEC 23001-17	ISO/IEC 23001-17

ISO/IEC 23001-17 provides flexible mechanisms for the carriage of uncompressed Class 0 Motion Imagery within ISO Base Media File Format. When producing subsampled YUV imagery, ISO/IEC 23001-17 supports 4:4:4, 4:2:2, 4:2:0, and 4:1:1 formats. The degrading image transformation of 4:4:4 to 4:2:2 or 4:2:0 results in a lossy uncompressed image as color information is averaged, resulting in unretrievable loss from the original. As the MISP does not allow the use of interlaced video, implementers must not use the 4:1:1 format.

11.3. Class 1 Motion Imagery

In the Class 1 Motion Imagery (MISP) we can find broadcast forms of compressed Motion Imagery, metadata, audio, and suitable containers, such as SD, HD, 4K, etc. It is applicable when delivering monochrome and color Motion Imagery in cases where the transmission bandwidth prohibits the use of Class 0 Motion Imagery. It allows for constrained image characteristics such as limits in pixel value range, number of bands, and number of images per second. Example is H.264/AVC compressed airborne Motion Imagery with three bands of color.

REQUIREMENTS CLASS 8: REQUIREMENTS CLASS 'CLASS 1 MOTION IMAGERY'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/class1-motion-imagery`
TARGET TYPE	GIMI File
PREREQUISITE	/req/motion-imagery

REQUIREMENTS CLASS 8: REQUIREMENTS CLASS 'CLASS 1 MOTION IMAGERY'	
NORMATIVE STATEMENT	Requirement 8-1: `/req/motion-imagery/codecs-class1`

Class 1 Motion Imagery applications may use the HEVC (ISO/IEC 23008-2) or AVC (ISO/IEC 14496-10) codecs, with Levels and Profiles as defined by the MISP.

REQUIREMENT 15	
IDENTIFIER	`/req/motion-imagery/codecs_class1`
A	Where a GIMI file contains motion imagery class 1, the coding of the motion imagery SHALL conform to one of the codec standards listed Table 6.

Table 6 — List of codecs approved for use with Class 1 Motion Imagery

Encoding Method	Codec Standard	Carriage in GIMI
HEVC/H.265	ISO/IEC 23008-2	ISO/IEC 14496-12, ISO/IEC 14496-15
AVC/H.264	ISO/IEC 14496-10	ISO/IEC 14496-12, ISO/IEC 14496-15

11.4. Class 2 Motion Imagery

In the Class 2 Motion Imagery (MISP) we can find compressed Motion Imagery, metadata, audio, and suitable containers, with unconstrained image characteristics. Unlike Class 1 Motion Imagery, Class 2 Motion Imagery allows for a non-constrained set of image characteristics. Examples include high frame rate scientific imaging, Large Volume Motion Imagery, and high bit-depth compressed infrared.

REQUIREMENTS CLASS 9: REQUIREMENTS CLASS 'CLASS 2 MOTION IMAGERY'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/class2-motion-imagery`
TARGET TYPE	GIMI File
PREREQUISITE	/req/motion-imagery

REQUIREMENTS CLASS 9: REQUIREMENTS CLASS 'CLASS 2 MOTION IMAGERY'	
NORMATIVE STATEMENT	Requirement 9-1: `/req/motion-imagery/codecs-class2`

Class 2 Motion Imagery may use the HEVC and AVC codecs.

REQUIREMENT 16	
IDENTIFIER	`/req/motion-imagery/codecs_class2`
A	Where a GIMI file contains motion imagery class 2, the coding of the motion imagery SHALL conform to one of the codec standards listed Table 7.

Table 7 — List of codecs approved for use with Class 2 Motion Imagery

Encoding Method	Codec Standard	Carriage in GIMI
HEVC/H.265	ISO/IEC 23008-2	ISO/IEC 14496-12, ISO/IEC 14496-15
AVC/H.264	ISO/IEC 14496-10	ISO/IEC 14496-12, ISO/IEC 14496-15

12

REQUIREMENT CLASSES FOR SECURITY MARKINGS

12 REQUIREMENT CLASSES FOR SECURITY MARKINGS

This requirements class specifies how to include NSG Security XML markings in a GIMI file. This conformance class is only mandatory for all NSG GIMI files. All other activities are exempt from the use of the 'sm01' brand and exempt from the inclusion of GIMI Security XML markings. For example, commercial applications using NGA.STND.0076-01 are not required to have security markings and should not carry a security marking brand.

REQUIREMENTS CLASS 10: REQUIREMENTS CLASS 'SECURITY MARKINGS'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/security-marking`
TARGET TYPE	GIMI File
PREREQUISITES	Requirements class 2: `https://www.opengis.net/spec/GIMI/1.0/req/content-id` Requirements class 3: `https://www.opengis.net/spec/GIMI/1.0/req/tai`
NORMATIVE STATEMENTS	Requirement 17: `/req/security-marking/sm01-brand` Requirement 18: `/req/security-marking/security-schema`

12.1. The sm01 brand

NGA.STND.0076-01 defines the 'sm01' brand to indicate the presence of GIMI Security XML containing U.S. Government ODNI ISM.XML security markings. This branding methodology and interpretation supports adding additional security marking brands to accommodate other forms of security markings (e.g., NATO), by creating additional brands (such as 'sm02', etc.).

REQUIREMENT 17	
IDENTIFIER	`/req/security-marking/sm01-brand`
INCLUDED IN	Requirements class 10: `https://www.opengis.net/spec/GIMI/1.0/req/security-marking`
A	A GIMI file including security markings SHALL include the `sm01` brand in the FileTypeBox (`ftyp`) in the compatible brands array.

12.2. The security marking format

REQUIREMENT 18	
IDENTIFIER	/req/security-marking/security-schema
INCLUDED IN	Requirements class 10: https://www.opengis.net/spec/GIMI/1.0/req/security-marking
A	A security marking in an NSG GIMI file SHALL be provided for all primary-media- content in the file.
B	A security marking in an NSG GIMI file SHALL be expressed in XML.
C	An XML security marking in an NSG GIMI file SHALL conform to the GIMI Security XML schema in Listing 1.
D	An Security XML “<File>” element declaration SHALL include the ism:resourceElement= ”true” statement.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources [
  <!ENTITY uuid_urn "[a-f0-9]{8}-([a-f0-9]{4}-){3}[a-f0-9]{12}">
]>
<?xml-model href="../ISM/Schematron/ISM/ISM_XML.sch"
            type="application/xml"
            schematypens="http://purl.oclc.org/dsdl/schematron"?>
<?xml-model href="../ISM/Schematron/ISMCAT/ISMCAT_XML.sch"
            type="application/xml"
            schematypens="http://purl.oclc.org/dsdl/schematron"?>
<!-- Notices - distEditionBlockReplace -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning"
   vc:minVersion="1.0" targetNamespace="urn:us:mil:nga:stnd:0076:ism"
   xmlns:ism="urn:us:gov:ic:ism" xmlns:arh="urn:us:gov:ic:arh"
   xmlns:ntk="urn:us:gov:ic:ntk"
xmlns:gimi="urn:us:mil:nga:stnd:0076:ism"
   ism:resourceElement="true"
   xmlns:xhtml="http://www.w3.org/1999/xhtml-StopBrowserRendering"
   ism:compliesWith="USGov" ism:createDate="2011-12-01"
   ism:ISMCATCESVersion="202405" ism:DESVersion="202405"
   ism:classification="U" ism:ownerProducer="USA" version="202405"
   elementFormDefault="qualified">
  <xs:import namespace="urn:us:gov:ic:ism"
      schemaLocation="../ISM/Schema/ISM/IC-ISM.xsd" />
   <xs:import namespace="urn:us:gov:ic:arh"
      schemaLocation="../ISM/Schema/ISM/IC-ARH.xsd" />
   <xs:import namespace="urn:us:gov:ic:ntk"
      schemaLocation="../ISM/Schema/ISM/IC-NTK.xsd" />
  <xs:element name="GIMISecurity">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="gimi:File" minOccurs="1" maxOccurs="1"/>
```

```
      </xs:sequence>
      <xs:attributeGroup ref="ism:ISMRootNodeAttributeGroup"/>
      <xs:attributeGroup ref="ism:ISMResourceAttributeOptionGroup"/>
      <xs:attribute name="GIMISecVer" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="File">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="arh:Security" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="gimi:Content" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Content">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>
          <xhtml:p ism:classification="U" ism:ownerProducer="USA">
            Lifted from ISM to avoid the questionable rule 511 about
            security elements being required to be resource node true.
          </xhtml:p>
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element ref="ntk:Access" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="ism:NoticeList" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="id" type="gimi:ContentIDType" use="required"/>
      <xs:attributeGroup ref="arh:ResourceNodeAttributeGroup"/>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="ContentIDType">
    <xs:restriction base="xs:string">
      <!-- Leading digit of guide group is not zero, max 15 digits -->
<xs:pattern value="urn:uuid:[a-f0-9]{8}-([a-f0-9]{4}-){3}[a-f0-9]
{12}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Listing 1 — GIMI Security XML Schema File

13

REQUIREMENT CLASSES FOR GEOREFERENCING BY AN RDF DOCUMENT

13 REQUIREMENT CLASSES FOR GEOREFERENCING BY AN RDF DOCUMENT

This requirement class describes how to add an RDF metadata document in a HEIF document. A HEIF writer needs to decide whether the RDF content is stored internal to a GIMI file or in an external resource, such as an RDF sidecar file.

13.1. Requirement Class for Georeferencing by an RDF document

REQUIREMENTS CLASS 11: REQUIREMENTS CLASS 'RDF METADATA'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/rdf-metadata`
TARGET TYPE	GIMI File
PREREQUISITES	Requirements class 2: `https://www.opengis.net/spec/GIMI/1.0/req/content-id` Requirements class 3: `https://www.opengis.net/spec/GIMI/1.0/req/tai`
NORMATIVE STATEMENT	Requirement 19: `/req/rdf-metadata/metadata-schema`

REQUIREMENT 19	
IDENTIFIER	`/req/rdf-metadata/metadata-schema`
INCLUDED IN	Requirements class 11: `https://www.opengis.net/spec/GIMI/1.0/req/rdf-metadata`
A	If the images in a GIMI file are georeferenced, an RDF metadata string SHALL be provided.
B	The RDF metadata in an NSG GIMI file SHALL be expressed in turtle.
C	For each georeferenced image, an RDF triple with *predicate* `cco:ont00001808` and *object* the `Content Id` of the image SHALL be included. A triple with the same *subject* and with *predicate* `geosparql:asWKT` SHALL be included with *object* a polygon encoded in WKT defining the bounding box of the image.

Assuming that there is an image that requires georeference, with a Content Id <urn:uuid:0a655014-e62a-0ead-5ae1-284b5a9304191932>

```
@prefix cco: <https://www.commoncoreontologies.org/> .
@prefix tb21: <http://www.opengis.net/ont/tb21/> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .

<urn:uuid:1a9a5434-4a81-b48b-97da-774f33c0758b488b>
    tb21:hasCorr <urn:uuid:1ef7b441-ee30-5530-5379-65d6d13d0dce36bf>, <urn:
uuid:489f6bd0-8425-c4c4-b867-1541a25c7927fbf8>, <urn:uuid:6f7451da-f2a4-391a-
02da-6ae322b856bc9656>, <urn:uuid:75c6a59e-0de9-bb22-3389-262c07e64a6caf0e> ;
    geosparql:asWKT "POLYGON((138.498668 -34.800998,138.498669 -
34.804844,138.494192 -34.800998,138.494192 -34.804844,138.498668 -
34.800998))"^^geosparql:wktLiteral ;
    a tb21:RasterBoundsCorrGroup, geosparql:geometry ;
    rdfs:label "Correspondence Group"^^xsd:string ;
    cco:ont00001808 <urn:uuid:0a655014-e62a-0ead-5ae1-284b5a9304191932> .
```

Listing 2 — Testbed-21 GIMI RDF turtle fragment example

13.2. Requirement Class for Georeferencing by an embedding an RDF document

This requirements class discusses the possibility of embedding the RDF documentation in the GIMI file. This requirements class is an alternative to the next requirements class.

REQUIREMENTS CLASS 12: REQUIREMENTS CLASS 'RDF METADATA IN THE FILE'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/rdf-metadata-in`
TARGET TYPE	GIMI File
PREREQUISITE	Requirements class 11: `https://www.opengis.net/spec/GIMI/1.0/req/rdf-metadata`
NORMATIVE STATEMENT	Requirement 20: `/req/rdf-metadata-in/in`

REQUIREMENT 20	
IDENTIFIER	`/req/rdf-metadata-in/in`
INCLUDED IN	Requirements class 12: `https://www.opengis.net/spec/GIMI/1.0/req/rdf-metadata-in`

REQUIREMENT 20	
A	If images in a GIMI file are georeferenced, an RDF document SHALL be embedded in the HEIF file.
B	A HEIF SHALL include an item with `itemType` with value 'mime' and `contentType` with value 'text/turtle' containing the RDF metadata.

13.3. Requirement Class for Georeferencing by a sidecar RDF document

This requirements class discusses the possibility to have an RDF document as a sidecar file with the GIMI file. This requirements class is an alternative to the previous requirements class.

REQUIREMENTS CLASS 13: REQUIREMENTS CLASS 'RDF METADATA IN SIDECAR'	
IDENTIFIER	`https://www.opengis.net/spec/GIMI/1.0/req/rdf-metadata-sidecar`
TARGET TYPE	GIMI File
PREREQUISITE	Requirements class 11: `https://www.opengis.net/spec/GIMI/1.0/req/rdf-metadata`
NORMATIVE STATEMENT	Requirement 21: `/req/rdf-metadata-sidecar/sidecar`

REQUIREMENT 21	
IDENTIFIER	`/req/rdf-metadata-sidecar/sidecar`
INCLUDED IN	Requirements class 13: `https://www.opengis.net/spec/GIMI/1.0/req/rdf-metadata-sidecar`
A	If images in a GIMI file are georeferenced, an RDF file containing the RDF-encoded metadata SHALL accompany the HEIF file.

14

MEDIA TYPES FOR ANY DATA ENCODING(S)

14 MEDIA TYPES FOR ANY DATA ENCODING(S)

This document does not specify any particular media type for GIMI. As specified in https://www.iana.org/assignments/media-types/image/heic, media files will have the media image/heic only if the file conforms to requirements of the 'heic', 'heix', 'heim', or 'heis' brand, and contains at least one of those brands as a compatible brand. The media type image/heif will be used only if the file conforms to the requirements of the 'mif1' brand, and contains that brand as a compatible brand.

File extensions are not a clear indication of the file content. Commonly, the file extension identifies the primary use of the file. For instance, files carrying both Still and Motion Imagery may use a file extension of `.heif` or `.mp4`. Reader applications utilize brands in the FileTypeBox to determine which specification(s) a file and its contents conforms to, regardless of the extension.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Joan Masó Pau, Núria Julià Selvas, Rob Smith, Open Geospatial Consortium: OGC 24-040r1, *OGC Testbed 20: GIMI Lessons Learned and Best Practices Report*. Open Geospatial Consortium (2025). http://www.opengis.net/doc/PER/t20-D013.

[2] International Organization for Standardization (committee): ISO/IEC 14496-15:2024, *Information technology — Coding of audio-visual objects — Part 15: Carriage of network abstraction layer (NAL) unit structured video in the ISO base media file format*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2024). https://www.iso.org/standard/89118.html.

[3] International Organization for Standardization (committee): ISO/IEC 15444-1:2024, *Information technology — JPEG 2000 image coding system — Part 1: Core coding system*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2024). https://www.iso.org/standard/87632.html.

[4] ISO/IEC: ISO/IEC 15444-16:2025, *Information technology -JPEG 2000 image coding system — Part 16: Enhanced encapsulation of JPEG 2000 images into ISO/IEC 14496-12.*. ISO, IEC (2025).

[5] International Organization for Standardization (committee): ISO/IEC 23001-17:2024, *Information technology — MPEG systems technologies — Part 17: Carriage of uncompressed video and images in ISO base media file format*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2024). https://www.iso.org/standard/82528.html.

A

ANNEX A (INFORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

A ANNEX A (INFORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

NOTE: Not completed for this report.

B

ANNEX B (INFORMATIVE) HEIF STRUCTURAL ENCODING DESCRIPTION

ANNEX B (INFORMATIVE) HEIF STRUCTURAL ENCODING DESCRIPTION

B.1. Introduction

This Annex provides informative background on how GeoHEIF is encoded in various file structures.

B.2. ISO Base Media File Format and HEIF File Format structure

The starting point of reading a HEIF file is to know how to read a box, as files are composed by a sequence of boxes.

Boxes are binary objects encoded in the file. Fields in objects are stored with the most significant byte first (commonly known as *network byte order* or *big-endian* format. When fields are smaller than a byte, the bits are assigned using the most significant bit first. For example, a field of two bits followed by a field of six bits has the two bits in the high order bits of the byte and the six bits afterwards.

Boxes can be extended but the basic structure starts with the same header structure, including a box `size` and `type`, then content according to the requirements of that `Box`. Standards are allowed to defined new box types with the condition of respecting the first fields in the box that are fixed. The fields that are fixed are defined by two supertypes of boxes:

- The minimum box (called “Box”); and
- The extended box (called “FullBox”) that is a common extension of a box.

B.2.1. How to read a box

There is no way to know if a box will be an extension of `Box` or an extension of `FullBox` other than through checking the box type (second field in the minimum box definition) and checking the standard document where the extended box was defined.

In practice that means that, as a developer, you will assume that all boxes are of the type “Box”. You will start by reading the first box in the file. You will read the first fields in order to know the `size` of the box and the `type` of the box. Then you will check the `type` depending on the type, you will know it and you will also know if it is a FullBox or a Box extension, as well as, which are the other fields in the box. If you do not know the `type` you may simply ignore the box by jumping to the next box. You will repeat the same process until you reach the end of the file (or the size is 0). Knowing the ‘type’ of a box is not so easy as boxes can be defined in many standard extensions.

B.2.2. Box structure

The `Box` basic object is defined in ISO/IEC 14496-12:2022 as:

```
aligned(8) class Box (unsigned int(32) boxtype,
    optional unsigned int(8)[16] extended_type) {
    unsigned int(32) size;
    unsigned int(32) type = boxtype;
    if (size==1)
        unsigned int(64) largesize;
    else if (size==0) {
        // box extends to end of file
    }
    if (boxtype==‘uuid’)
        unsigned int(8)[16] usertype = extended_type;
}
```

Listing B.1

Even if type if defined as an unsigned int(32) it is commonly transformed into a four character code (“4CC”), that is easy to remember and identify.

B.2.3. Full Box

The `FullBox` basic object is defined in ISO/IEC 14496-12:2022 as:

```
aligned(8) class FullBox(unsigned int(32) boxtype, unsigned int(8) v, bit(24) f)
    extends Box(boxtype) {
    unsigned int(8) version = v;
    bit(24) flags = f;
}
```

Listing B.2

This adds a version and flags to the box that conditions the size and type that the `Box` had. This allows for changing the object fields depending on the versions and flags.

B.2.4. Boxes data structure

The ISO Base Media File Format (ISOBMFF) is normatively described in ISO/IEC 14496-12:2022.

The box type is shown as `unsigned int(32) boxtype` in the box definition, but in fact they are a four character code ("4CC"). This four character code is mapped to a *box type name* and defined in some derived standard.

For that example, the unsigned 32 bit integer type code can read 0×6d646174 that corresponds to the ASCII encoding of `mdat`). The `mdat` box type is defined by a `MediaDataBox` structure is given as:

```
aligned (8) class MediaDataBox extends Box('mdat') {
    bit(8) data[];
}
```

Listing B.3

The High Efficiency Image File Format (HEIF) is normatively described in ISO/IEC 23008-12:2022. It builds on and profiles ISOBMFF to provide support for images and image sequences. This standard defined some specific boxes that are useful for images. This boxes together provide all information to read the actual image code that are commonly in the mdata box. For images, an indicative structure is shown below.

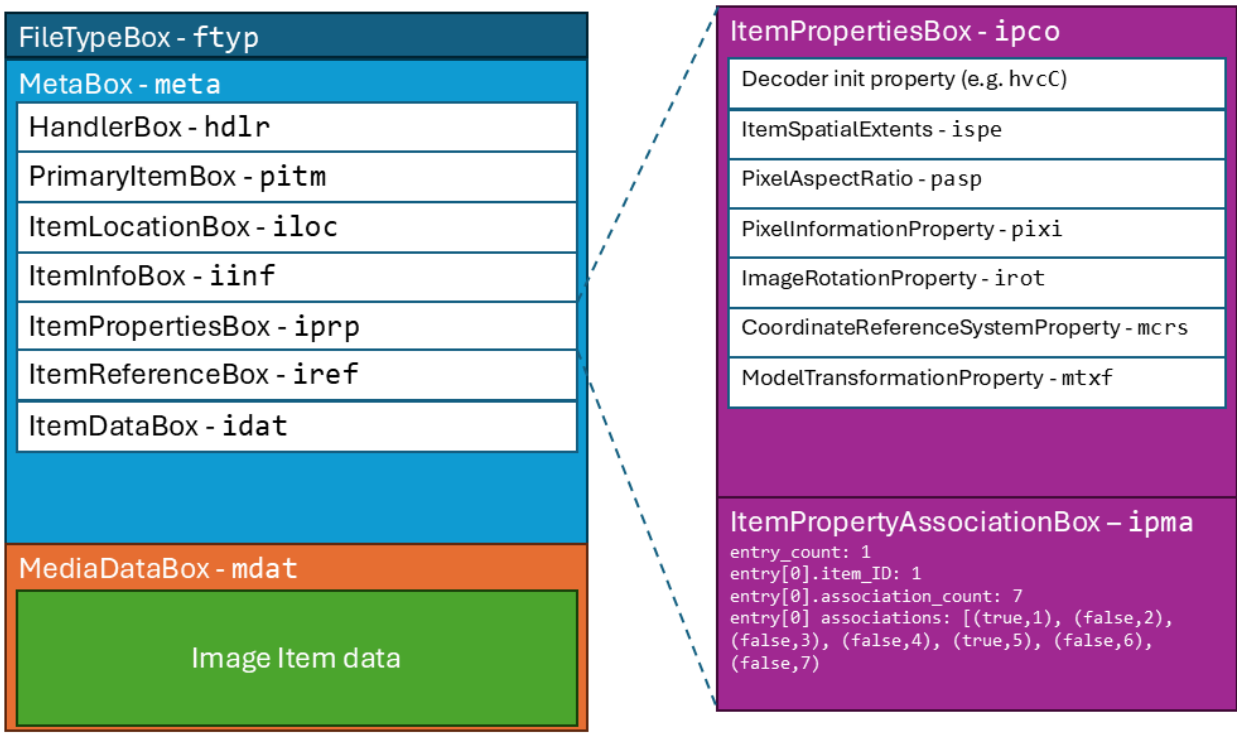

Figure B.1 — Indicative HEIF file structure

As shown in this example, some boxes can be nested. Typically, the body of the outer box is just a sequence of inner boxes. This is the case for the MetaBox (`meta`) in the figure above.

Now we are going to review the most relevant boxes for images.

B.2.5. FileTypeBox and brands

The FileTypeBox (`ftyp`) identifies the file structure as ISOBMFF, and what functionality is required to be able to interpret the file.

The `FileTypeBox` object is defined in ISO/IEC 14496-12:2022 as:

```
aligned(8) class FileTypeBox
    extends Box('ftyp') {
    unsigned int(32) major_brand;
    unsigned int(32) minor_version;
    unsigned int(32) compatible_brands[]; // to end of the box
}
```

Listing B.4

The body of the FileTypeBox contains the major brand and an array of compatible brands, which are four character codes (4CC) encoded in the same way as the box type. A typical brand for an image file is `mif1`.

The meaning of the different brands may be a bit difficult to grasp, as new brands can be defined in new file format extensions.

We have collected some possible brands in the following table:

Table B.1

BRAND	MEANING
'mif1'	HEIF File: Still imagery version 1.
'mif2'	HEIF File: Still imagery version 2"
'msf1'	HEIF File: Image sequence version 1"
'msf2'	HEIF File: Image sequence version 2"
'unif'	Unified handling of IDs across file scoped MetaBox items, tracks, track groups, and entity groups.
'geo1'	GIMI File: conformant to version 1 of NGA.STND.0076_1.0
'sm01'	GIMI File with Security XML markings

In the OGC Testbed 20 there was a proposal to brand GeoHEIF files as `ogeo` brand to mark the file as “geospatial reference — either georeferenced or georeferenceable” but this proposal was abandoned.

B.2.6. MetaBox

The `MetaBox` object (4CC type `meta`) is defined in ISO/IEC 14496-12:2022 as:

```
aligned(8) class MetaBox (handler_type)
    extends FullBox('meta', version = 0, 0) {
    HandlerBox(handler_type) theHandler;
    PrimaryItemBox primary_resource; // optional
    DataInformationBox file_locations; // optional
    ItemLocationBox item_locations; // optional
    ItemProtectionBox protections; // optional
    ItemInfoBox item_infos; // optional
    IPMPControlBox IPMP_control; // optional
    ItemReferenceBox item_refs; // optional
    ItemDataBox item_data; // optional
    Box other_boxes[]; // optional
}
```

Listing B.5

This means that the meta box is only a sequence of other boxes (all different) being the HandlerBox the first and the only mandatory one.

B.2.7. HandlerBox

The `HandlerBox` object (4CC type `hdlr`) is the first one found in the MetaBox. It is defined in ISO/IEC 14496-12:2022 as:

```
aligned(8) class HandlerBox extends FullBox('hdlr', version = 0, 0) {
    unsigned int(32) pre_defined = 0;
    unsigned int(32) handler_type;
    const unsigned int(32)[3] reserved = 0;
    string name;
}
```

Listing B.6

It provides a handler type that is expressed as a four character code (4CC) encoded in the same way as the box type. A handler type for an image file is `pict`.

Some possible brands are presented in the following table:

Table B.2

HANDLER TYPE	MEANING	DEFINED
‘pict’	Picture media	ISO23008-12

HANDLER TYPE	MEANING	DEFINED
‘vide’	Video media	ISO14496-12
‘soud’	Sound media	ISO14496-12
‘text’	Text media	ISO14496-12

B.2.8. Items in an HEIF file

To allow reader applications to determine what information is present in the HEIF file, along with where the data is located and how to interpret that data, there is supporting structure of boxes.

For timed data (e.g., video tracks, audio tracks, and associated metadata and subtitles), this is provided in the MovieBox (`moov`), discussed later in this Appendix.

For image data, the MetaBox (`meta`) provides the entry point to the different data elements that can be found in the file. Some of these elements can represent images. These elements are called `items`. The MetaBox is a container for some nested boxes including:

- the ItemInfoBox, which identifies the various items that are present,
- the ItemLocationBox, which informs about the location of the encoded data (often by byte offsets into the file), and
- the ItemPropertiesBox, which provides descriptive (metadata) and transformative properties.

Item properties generally describe how to interpret the encoded data to reconstruct the required image. Transformative properties change the resulting image, and include operations such as mirroring, rotation and scaling. Descriptive properties provide information that is either required or optional for reconstruction. If a property is required (such as the codec configuration), it is marked as essential.

B.2.9. ItemInfoBox and ItemInfoEntries

The `ItemInfoBox` object (4CC type `iinf`) is defined in ISO/IEC 14496-12:2022 as:

```
aligned(8) class ItemInfoBox
    extends FullBox(‘iinf’, version, 0) {
    if (version == 0)
        unsigned int(16) entry_count;
    else
        unsigned int(32) entry_count;
    ItemInfoEntry[entry_count] item_infos;
}
```

Listing B.7

This object is an array of entry_count ItemInfoEnties. The ItemInfoEntries are commonly called `items` represent “things” that can be read an extracted from the file. Items can be textual strings, images, grid tiles, metadata in RDF etc. However, items themselves contain only a few metadata fields and have not enough information to read the actual content (commonly stored on the MediaDataBox) and a library trying to understand the format will need to read other boxes in the MetaBox before it is able to finally find and extract the data in the ‘mdat’ (MediaDataBox).

An ItemInfoEntry is an object (4CC type `infe`) defined in ISO/IEC 14496-12:2022 as:

```
aligned(8) class ItemInfoEntry
    extends FullBox(‘infe’, version, 0) {
    if ((version == 0) || (version == 1)) {
        unsigned int(16) item_ID;
        unsigned int(16) item_protection_index
        string item_name;
        string content_type;
        string content_encoding; //optional
        if (version == 1) {
            unsigned int(32) extension_type; //optional
            ItemInfoExtension(extension_type); //optional
        }
    }
    else { // if (version >= 2)
        if (version == 2)
            unsigned int(16) item_ID;
        else if (version == 3)
            unsigned int(32) item_ID;

        unsigned int(16) item_protection_index;
        unsigned int(32) item_type;

        string item_name;
        if (item_type=='mime') {
            string content_type;
            string content_encoding; //optional
        } else if (item_type == 'uri')
            string item_uri_type;
    }
}
```

Listing B.8

The item box (ItemInfoEntry) looks complicated to read but, in essence, each `item` has an `id`, a `type` and eventually a `media type`. The item `id` is the most important field, as it will be used by other boxes to point to this item and add metadata or inform about the offsets to the item.

B.2.10. ItemPropertiesBox, properties and ItemPropertyAssociation

A common way to add extra properties to an ‘item’ is by reading the ItemPropertiesBox object (4CC type `iprp`) defined in ISO/IEC 23008-12:2022.

```
aligned(8) class ItemPropertiesBox
    extends Box('iprp') {
    ItemPropertyContainerBox property_container;
    ItemPropertyAssociation association[];
```

```
}
```

Listing B.9

This box contains two boxes: a list of properties (in a ItemPropertyContainerBox object (4CC type `ipco`)) and the mechanism to associate properties to items (in a ItemPropertyAssociation object (4CC type `ipma`))

The item properties are contained in the ItemPropertyContainerBox, concatenated one after another. The item properties are associated with the image item by an entry in the ItemPropertyAssociationBox, which provides the property indexes for each property that apply to an item, and whether the property is essential for the consumer to understand (i.e., the `true` part in the associations array).

NOTE 1: Property indexes are 1-based.

A ItemPropertyContainerBox object is defined as:

```
aligned(8) class ItemPropertyContainerBox
    extends Box('ipco')
{
    properties ItemProperty() []; // boxes derived
}
```

Listing B.10

ItemProperties() is an "abstract box" that can be boxes extended from `Box` (ItemProperty) or `FullBox` (ItemFullProperty). A library trying to read these boxes will start by finding out the size and type of the boxes (as all boxes starts in the same way), and then interpreting the properties it knows or jumping over the ones it does not understand (the latter happens often as other standards can defined more properties as extensions). A following section will detail some of the properties that need to be read to understand the type of images described in this document.

Once the properties are read, we are in position to read the associations. This is defined by ItemPropertyAssociation object (4CC type `ipma`) defined in ISO/IEC 23008-12:2022.

```
aligned(8) class ItemPropertyAssociation
    extends FullBox('ipma', version, flags)
{
    unsigned int(32) entry_count;
    for(i = 0; i < entry_count; i++) {
        if (version < 1)
            unsigned int(16)   item_ID;
        else
            unsigned int(32)   item_ID;
        unsigned int(8) association_count;
        for (i=0; i<association_count; i++) {
            bit(1) essential;
            if (flags & 1)
                unsigned int(15) property_index;
            else
                unsigned int(7) property_index;
        }
    }
}
```

Listing B.11

This means that an item `id` can be associated to one or more properties by `index`. From the point of view of a library reading items, the properties will be added as metadata to the item. None of these properties are the location of the actual content, that is specified in a different box.

NOTE 2: Properties in the ItemPropertyContainerBox do not have to be used by any item.

The way the association works is optimized for re-use of properties where there is more than one item in the file. For example, a `grid` image type works by linking to multiple items, which can share the same decoder configuration and image spatial extent (`ispe` — the pixel space width and height) properties.

B.2.11. Item content and the ItemLocationBox

The encoded data in ISOBMFF can be contained in a range of locations. The usual location is in one or more MediaDataBox (`mdat`) instances in the file. Within the `mdat`, the bytes are potentially un-ordered arrays. The `mdat` does not contain information about the position of the content (with some exceptions, including the `tili` approach). Instead, the ItemLocationBox provides information on the offsets of the content of the items in the `mdat` box.

This is defined by ItemLocationBox object (4CC type `iloc`) defined in ISO/IEC 14496-12:2022.

```
aligned(8) class ItemLocationBox extends FullBox('iloc', version, 0) {
    unsigned int(4) offset_size;
    unsigned int(4) length_size;
    unsigned int(4) base_offset_size;
    if ((version == 1) || (version == 2))
        unsigned int(4) index_size;
    else
        unsigned int(4) reserved;

    if (version < 2)
        unsigned int(16) item_count;
    else if (version == 2)
        unsigned int(32) item_count;

    for (i=0; i<item_count; i++) {
        if (version < 2)
            unsigned int(16) item_ID;
        else if (version == 2)
            unsigned int(32) item_ID;

        if ((version == 1) || (version == 2)) {
            unsigned int(12) reserved = 0;
            unsigned int(4) construction_method;
        }
        unsigned int(16) data_reference_index;
        unsigned int(base_offset_size*8) base_offset;
        unsigned int(16) extent_count;
        for (j=0; j<extent_count; j++) {
            if (((version == 1) || (version == 2)) && (index_size > 0))
                unsigned int(index_size*8) extent_index;
            unsigned int(offset_size*8) extent_offset;
            unsigned int(length_size*8) extent_length;
        }
    }
```

```
}
```

Listing B.12

The `item_ID` indicates the identifier of the item the location of the content of which will be described. There are three methods for constructing the offsets, being the most simple the `construction_method`=0. In these cases the base_offset is added to the extent_offset to know the actual position of the data. For each item id, the data can be fragments in several pieces (as many as `extent_count`); so the offsets and lengths are an array, but this is only present if the data is interleaved.

NOTE 1: the word 'extent' is used here in the sense of 'offset' of the content and is not related to the 'geospatial extent'.

NOTE 2: Other locations that may be used include the ItemDataBox (`idat`) nested within the MetaBox (`meta`), an IdentifiedMediaDataBox (`imda`) or in an external file. These are less common and may have less support within reader software.

B.2.12. Boxes in the array of ItemPropertyBox

B.2.12.1. ImageSpatialExtentsProperty

The ImageSpatialExtentsProperty is one of the most fundamental boxes for still imagery. The object (4CC type `ispe`) is defined in ISO/IEC 23008-12:2022.

This is defined as:

```
aligned(8) class ImageSpatialExtentsProperty
extends ItemFullProperty('ispe', version = 0, flags = 0) {
    unsigned int(32) image_width;
    unsigned int(32) image_height;
}
```

Listing B.13

It indicates the width and height of the image of the associated item id in pixel units.

NOTE: The expression "Spatial Extent" is used in this context to refer to refer to the width and height of the image. The meaning is not related to the 'geospatial extent'.

B.2.12.2. PixelInformationProperty

The PixelInformationProperty is one of the most fundamental boxes for still imagery. The object (4CC type `pixi`) is defined in ISO/IEC 23008-12:2022.

This is defined as:

```
aligned(8) class PixelInformationProperty
extends ItemFullProperty('pixi', version = 0, flags = 0){
    unsigned int (8) num_channels;
    for (i=0; i<num_channels; i++) {
```

```
        unsigned int (8) bits_per_channel;
    }
}
```

Listing B.14

It indicates the number of bits per channel and pixel of the image of the associated item id.

B.2.12.3. ColourInformationBox

The ColourInformationBox is a box that defines how the colors of the pixels in an image or video should be interpreted. The object (4CC type `colr`) is defined in ISO/IEC 14496-12:2022.

This is defined as:

```
class ColourInformationBox extends Box('colr'){
    unsigned int(32) colour_type;
    if (colour_type == 'nclx') /* on-screen colours */
    {
        unsigned int(16) colour_primaries;
        unsigned int(16) transfer_characteristics;
        unsigned int(16) matrix_coefficients;
        unsigned int(1) full_range_flag;
        unsigned int(7) reserved = 0;
    }
    else if (colour_type == 'rICC')
    {
        ICC_profile; // restricted ICC profile
    }
    else if (colour_type == 'prof')
    {
        ICC_profile; // unrestricted ICC profile
    }
}
```

Listing B.15

B.2.12.4. HEVCDecoderConfigProperty

The HEVCDecoderConfigurationProperty is a box that describes some properties for a still image in HEVC format. The object (4CC type `hvcC`) is defined in ISO/IEC 14496-15:2024.

This is defined as:

```
aligned (8) class HEVCDecoderConfigProperty
extends ItemFullProperty('hvcC', version = 0, flags = 0){
    unsigned int(8) configurationVersion = 1
    unsigned int(2) general_profile_space;
    unsigned int(1) general_tier_flag;
    unsigned int(5) general_profile_idc;
    unsigned int(32) general_profile_compatibility_flags;
    unsigned int(48) general_constraint_indicator_flags;
    unsigned int(8) general_level_idc;
    bit(4) reserved='1111'b;
    unsigned int(12) min_spatial_segmentation_idc;
    bit(6) reserved='111111'b;
    unsigned int(2) parallelism_type;
```

```
    bit(5) reserved='11111'b;
    unsigned int(3) bit_depth_luma_minus8;
    bit(5) reserved='11111'b;
    unsigned int(3) bit_depth_chroma_minus8;
    unsigned int(16) avg_frame_rate;
    unsigned int(2) constant_frame_rate;
    unsigned int(3) num_temporal_layers;
    unsigned int(1) temporal id_nested;
    unsigned int(2) length_size_minus1;
    unsigned int(8) num_of_arrays;
    for (j=0; j < num_of_arrays; j++) {
        unsigned int(1) array_completeness;
        bit (1) reserved = 0;
        unsigned int(6) NAL_unit_type;
        unsigned int (16) num_nalus;
        for (i=0 i<num_nalus; i++) (
        unsigned int(16) nal_unit_length;
        bit(8*nal_unit_length) nal_unit;
    }
}
```

Listing B.16

B.2.12.5. UncompressedFrameConfigProperty

The UncompressedFrameConfigProperty is a box that describes some properties for still imagery in uncompressed format. The object (4CC type `uncC`) is defined in ISO/IEC 23001-17:2024.

This is defined as:

```
aligned(8) class UncompressedFrameConfigProperty
extends ItemFullProperty('uncC', version = 0, flags = 0) {

    unsigned int(32) uncompress_profile;
 if (version==1) {
        ;
    }
    else if (version==0) {
        unsigned int(32) component_count;
        {
            unsigned int(16) component_index;   //should be 32 bits !!
            unsigned int(8) component_bit_depth_minus_one;
            unsigned int(8) component_format;
            unsigned int(8) component_align_size;
        } [component_count];

        unsigned int(8) sampling_type;
        unsigned int(8) interleave_type;
        unsigned int(8) block_size;
        bit(1) components_little_endian;
        bit(1) block_pad_lsb;
        bit(1) block_little_endian;
        bit(1) block_reversed;
        bit(1) pad_unknown;
        bit(3) reserved = 0;

        unsigned int(32) pixel_size;
        unsigned int(32) row_align_size;
        unsigned int(32) tile_align_size;
        unsigned int(32) num_tile_cols_minus_one;
```

```
        unsigned int(32) num_tile_rows_minus_one;
    }
}
```

Listing B.17

The “uncompress_profile” is a character code (4CC) encoded in the same way as the box type. Value 0 indicates this uncompressed frame configuration may or may not be compliant to a profile. If not 0, all remaining fields in the box shall have the values indicated in the definition of this profile.

B.2.12.6. ComponentDefinitionBox/Property

The ComponentDefinitionBox is a box that describes the types of components present in samples or items in uncompressed format. The object (4CC type `cmpd`) is defined in ISO/IEC 23001-17:2024. This is defined as:

```
aligned(8) class ComponentDefinitionBox extends Box('cmpd') {
 unsigned int(32) component_count;
 {
  unsigned int(16) component_type;
  if (component_type>=0x8000) {
   utf8string component_type_uri;
  }
 } [component_count];
}
```

Listing B.18

B.2.12.7. CompressionConfigurationBox

The CompressionConfigurationBox is a box that specifies the data compression method used within a media sample or item data in uncompressed format. The object (4CC type `cmpC`) is defined in ISO/IEC 23001-17:2024. This is defined as:

```
aligned(8) class CompressionConfigurationBox extends FullBox('cmpC', version =
0, flags = 0) {

    unsigned int(32) compression_type;
    bit(1) must_decompress_individual_entities can_decompress_contiguous_ranges;
    bit(7) compressed_range_typecompressed_entity_type;
}
```

Listing B.19

B.2.12.8. JPEG 2000 header item property

The JPEG 2000 header item property is a box that contains several boxes describing still imagery in JPEG2000 format. The object (4CC type `j2kH`) is defined in ISO/IEC 15444-16:2025 as a set of subboxes.

This is defined as:

```
aligned(8) class ConfigProperty
extends ItemFullProperty('j2kH', version = 0, flags = 0) {
    J2KChannelDefinition channels[1];
    J2KComponentMapping components[0..1];
    J2KPalette palette[0..1];
    J2KLayers layers[0..1];
    J2KColour colour[0..*];
    J2KPixelFormat pixel_format[0..1];
}
```

Listing B.20

The actual content of the subboxes are defined in ISO/IEC 15444-1:2024 (2016)

One of the boxes is the J2KChannelDefinition (4CC type `j2kH`).

```
aligned(8) class J2KChannelDefinition
extends Box('cdef') {
    unsigned int(16) n_comp;
    for (i=0; i < n_comp; i++) {
        unsigned int(16) index;
        unsigned int(16) type;
        unsigned int(16) association;
    }
}
```

Listing B.21

The `type` parameter indicates the function of the component of the image. 0 means a color channel, 1 means opacity (0 meaning transparent and the max value is completely opaque), 2 means premultiplied opacity of the association channel.

C

ANNEX C (INFORMATIVE) GEOHEIF TILES ENCODING DESCRIPTION

ANNEX C
(INFORMATIVE)
GEOHEIF TILES ENCODING DESCRIPTION

Tiles and overviews are two important characteristics to improve the speed of data visualization in imagery. The approach is successfully used in Cloud Optimized GeoTIFF and can be reused in to generate a Seek Optimized HEIF (SOH). This annex discusses how to do tiles and overviews in HEIF without entering into the matter of how to exactly structure a Seek Optimized HEIF.

C.1. GroupsListBox

To find a collection of tiles and overviews among the items of a HEIF file, we start by finding the pyramid. This is defined as a group, so the first thing to do is reading a the box describing the groups. The GroupsListBox box is box that describes the groups. The object (4CC type `grpl`) is defined in ISO/IEC 14496-12:2022.

```
aligned(8) class GroupsListBox extends Box('grpl')
{
    entityToGroups EntityToGroupBox() []; // boxes derived
}
```

Listing C.1

So a GroupsListBox is only a list of EntityToGroupBox fullboxes

```
aligned(8) class EntityToGroupBox(grouping_type, version, flags)
 extends FullBox(grouping_type, version, flags)
{
 unsigned int(32) group_id;
 unsigned int(32) num_entities_in_group;
 for (i=0; i<num_entities_in_group; i++) {
 unsigned int(32) entity_id;
 }
 // the remaining data may be specified for a particular grouping_type
}
```

Listing C.2

It is just a list of entities that are group together. Entities are itemIDs in this case.

C.2. ImagePyramidEntityGroup

In a ImagePyramidEntityGroup each entity_id in the array of entities_in_group point to one level of the pyramid (a.k.a an overview in the COG language).

ImagePyramidEntityGroup is one of the possible EntityToGroupBox that extends EntityToGroupBox to define some additional properties defined in this pyramid level (called “layer” in the 23008-12 standard). The object (4CC type `pymd`) is defined in ISO/IEC 23008-12:2022 amendment 2.20.

```
aligned(8) class ImagePyramidEntityGroup
extends EntityToGroupBox ('pymd', version = 0, flags = 0) {
 unsigned int(16) tile_size_x;
 unsigned int(16) tile_size_y;
 for(i=0; i<num_entities_in_group;i++) {
  unsigned int(16) layer_binning;
  unsigned int(16) tiles_in_layer_row_minus1;
  unsigned int(16) tiles_in_layer_column_minus1;
 }
}
```

Listing C.3

In essence, it gives us the tile size in pixels (in here called tile_size_x and tile_size_y and in the 2D TileMatrixSet called tileWidth and TileHeight), the number of tiles in a row and in a column for this level or the pyramid (i here represented by tiles_in_layer_row_minus1 and tiles_in_layer_column_minus1 and in the 2D TileMatrixSet called MatrixHeight and MatrixWidth).

C.3. Actual tiles

If one level in the pyramid is represented by only one tile, the entity in group with point directly to and itemID that represents an image. There are another two cases

C.3.1. Grid tiles

If the number of tiles is relatively low, the grid approach can be used. An image item is fragmented into tiles and this grid tiles are other items. In order to know how the grid “fragments” belong to the image item representing the complete image, the ItemReferenceBox is used.

NOTE: The ‘grid’ structure requires some extra space (because each tile is defined as an item with its own properties). To maximize compression, ’tili’ is recommended over ‘grid’. However,

to maximize backwards compatibility with clients that does not understand 'tili', 'grid' is recommended when the total number of tiles permits it.

The ItemReferenceBox is part of the Metabox and is only an array of SingleItemTypeReferenceBox or SingleItemTypeReferenceBoxLarge. The object (4CC type `iref`) is defined in ISO/IEC 14496-12:2022.

```
aligned(8) class ItemReferenceBox extends FullBox('iref', version, 0)
{
    if (version==0) {
        SingleItemTypeReferenceBox references[];
    } else if (version==1) {
        SingleItemTypeReferenceBoxLarge references[];
    }
}
```

Listing C.4

Each SingleItemTypeReferenceBox or SingleItemTypeReferenceBoxLarge is a box that links one from_item_ID to one or more to_item_ID.

```
aligned(8) class SingleItemTypeReferenceBox(referenceType)
 extends Box(referenceType)
{
    unsigned int(16) from_item_ID;
    unsigned int(16) reference_count;
    for (j=0; j<reference_count; j++) {
        unsigned int(16) to_item_ID;
    }
}
aligned(8) class SingleItemTypeReferenceBoxLarge(referenceType)
 extends Box(referenceType)
{
    unsigned int(32) from_item_ID;
    unsigned int(16) reference_count;
    for (j=0; j<reference_count; j++) {
        unsigned int(32) to_item_ID;
    }
}
```

Listing C.5

As any other box, the SingleItemTypeReferenceBox or SingleItemTypeReferenceBoxLarge also has a type . If the type is 'dimg' and the fromItemId points to an item of itemType=`grid`, then

aligned(8) class ImageGrid { unsigned int(8) version = 0; unsigned int(8) flags; FieldLength = ((flags & 1) + 1) * 16; unsigned int(8) rows_minus_one; unsigned int(8) columns_minus_one; unsigned int(FieldLength) output_width; unsigned int(FieldLength) output_height; }

C.3.1.1. TiledImageConfigurationBox

The TiledImageConfigurationBox item property is box that describes the tili approach to tiles. The object (4CC type `tilC`) is defined in OGC 24-040r1 Annex B.

```
aligned(8) class TiledImageConfigurationBox
extends ItemFullProperty('tilC', version=0, flags) {
    unsigned int(32) tile_width;
```

```
    unsigned int(32) tile_height;
    unsigned int(32) tile_compression_type;
    unsigned int(8) number_of_extra_dimensions;
    for (int i=0; i<number_of_extra_dimensions; i++)
        unsigned int(32) dimension_size[i];

    unsigned int(8) number_of_tile_properties;
    for (int i=0; i<number_of_tile_properties; i++)
        ItemProperty tile_image_property[i];

}
```

Listing C.6

There are three extra attributes derived from flags:

```
OFFS_LEN[] = [ 32, 40, 48, 64 ]
offset_field_length = OFFS_LEN[flags & 0x03] //number of bits used to store the
offset to the image data of a specific tile.

SIZE_LEN[] = [ 0, 24, 32, 64 ]
size_field_length = SIZE_LEN[(flags>>2) & 0x03] //number of bits used to store
the length of the image data of a specific tile.

image_interlieved=(flags & 0x10)
```

Listing C.7

C.3.2. `tili` Item Data

The item data in the MetaBox pointed from the `iloc` box consists of an offset pointer table `TiledImageOffsetTable`, followed by the image data.

The number of tile offsets stored in the table (`num_tiles`) is computed by:

```
  matrix_width = (ispe_width + tile_width -1)/tile_width;
  matrix_height = (ispe_height + tile_height -1)/tile_height;

  num_tiles = matrix_width * matrix_height;
  for (i=0; i<number_of_extra_dimensions; i++)
    num_tiles = num_tiles * dimension_size[i];
```

Listing C.8

`ispe_width` and `ispe_height` is the total image size as specified in the mandatory `ispe` item property.

This is the composition of the TiledImageOffsetTable section (this is not a box so it has no header)

```
aligned(8) class TiledImageOffsetTable {

  for (int i=0; i<num_tiles; i++) {
    unsigned int(offset_field_length) tile_start_offset[i];
    unsigned int(size_field_length) tile_size[i];         // not present if size_
field_length==0
  }
}
```

```
// ... followed by compressed tile data ...
```

Listing C.9

The tile_start_offset[] point to the start of the data in the very same area. tile_start_offset[i] is relative to the start of the TiledImageOffsetTable data (neither to the start of the file nor the MediaDataBox).

D

ANNEX D (INFORMATIVE) TAI STRUCTURAL ENCODING DESCRIPTION

ANNEX D
(INFORMATIVE)
TAI STRUCTURAL ENCODING DESCRIPTION

D.1. Introduction

This Annex provides informative background on how TAI information is encoded in file structures.

Two data structures support TAI timestamps:

- TAIClockInfoBox: describes the TAI clock generating timestamps.
- TAITimestampPacket: contains a timestamp and quality status for the timestamp.

D.2. TAI Clock Info Box

The TAIClockInfoBox provides metadata about a specific TAI clock referenced by one or more TAITimestampPackets. This metadata includes information about the type, quality, and status of the TAI clock. This enables readers to accurately correlate information from multiple cameras or other sensors, for example temporally aligning frames between two disparate cameras with different clock sources.

The `TAIClockInfoBox` basic object is defined in ISO/IEC 23001-17:2024 Amd1 Annex D as:

```
aligned(8) class TAIClockInfoBox extends FullBox('taic', 0, 0) {{
    unsigned int(64) time_uncertainty;
    unsigned int(32) clock_resolution;
    signed int(32) clock_drift_rate;
    unsigned int(2) clock_type;
    unsigned int(6) reserved = 0;
}
```

Listing D.1

- `time_uncertainty` is the standard deviation measurement uncertainty for the timestamp , expressed as an unsigned integer number of nanoseconds.
- `clock_resolution` specifies the resolution of the receptor clock in nanoseconds.
- `clock_drift_rate` represents the rate of divergence of a receptor clock and its source of time when synchronization stops.
- `clock_type` indicates the synchronization capability values are de following:

Table D.1 — Synchronization capability

0	Clock type is unknown
1	The clock does not synchronize to an atomic source of absolute TAI time (e.g., unsynchronized CPU clock)
2	The clock can synchronize to an atomic source of absolute TAI time (e.g., synchronized GPS timing card)
3	Reserved

D.3. TAI timestamp packet

The `TAITimestampPacket` basic object is defined in ISO/IEC 23001-17:2024 Amd1 Annex D as:

```
aligned(8) class TAITimestampPacket {
    unsigned int(64) TAI_timestamp;
    unsigned int(1) synchronization_state;
    unsigned int(1) timestamp_generation_failure;
    unsigned int(1) timestamp_is_modified;
    unsigned int(5) reserved = 0;
}
```

Listing D.2

- `TAI_timestamp` is a 64-bit unsigned integer representing the number of nanoseconds since the TAI epoch of 1958-01-01T00:00:00.0.
- `synchronization_state` indicates the remote and receptor TAI clocks are in synchronization when creating the TAI timestamp.
- `timestamp_generation_failure` has a value of 1 when a TAI receptor clock does not report a TAI_timestamp.
- `timestamp_is_modified` signals when an existing TAI_timestamp is modified (e.g., modification to better align its value to the actual measurement time)

D.4. Item TAI timestamps

This is how to use a TAI as a item property:

```
aligned(8) class TAITimestampBox extends ItemFullProperty('itai', 0, 0) {
    TAITimestampPacket timestamp_packet;
}
```

Listing D.3