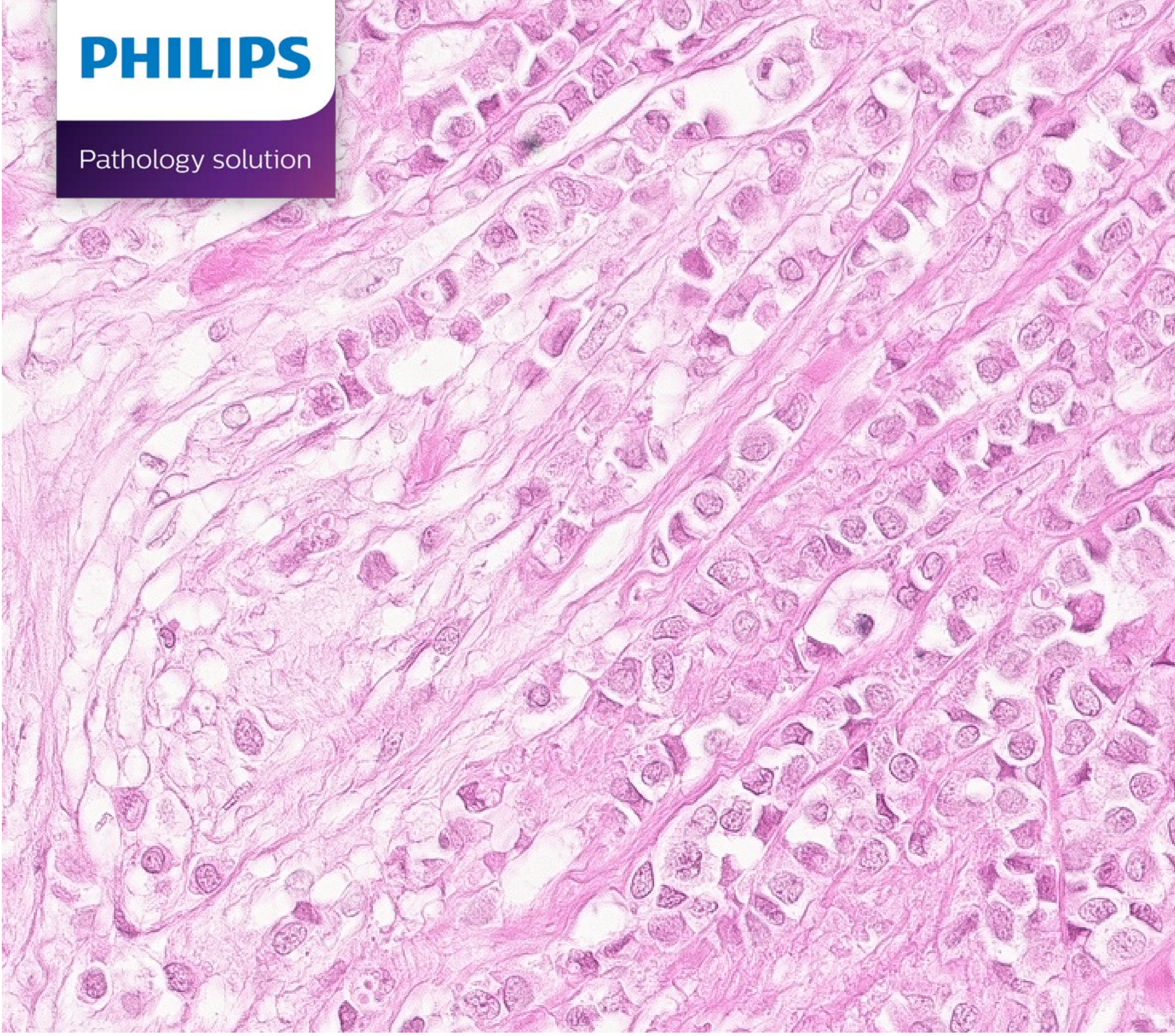

Pathology solution

Philips' iSyntax for Digital Pathology

Image format

1 Introduction

Digital pathology requires large amounts of gigapixel images to be generated, stored, and delivered with medical grade image quality and high performance to provide a seamless digital workflow. Philips uses the iSyntax format, which is leveraging Philips' leading IntelliSpace's iSyntax image representation for radiology images. The iSyntax format has distinguished features for storing pathology Whole Slide Images (WSI).

Philips is committed to an open pathology platform, enabling pathologists and researchers to unlock the power of digital pathology using Philips IntelliSite Pathology Solution (PIPS). All information and resources about the iSyntax format can be found on the Open Pathology Portal at www.openpathology.philips.com.

Image pipeline overview

The image pipeline utilized in Philips' solutions for digital pathology such as PIPS is built on iSyntax. The pipeline encompasses all the steps from creating and storing WSI data when scanning to displaying them to users. The following three steps are the main parts of the iSyntax image pipeline:

1. Write – compression of data in the iSyntax format and storing it
2. Read – reading of iSyntax data and decompressing it to create a source image
3. Post processing – processing of the source image to optimize for display to users

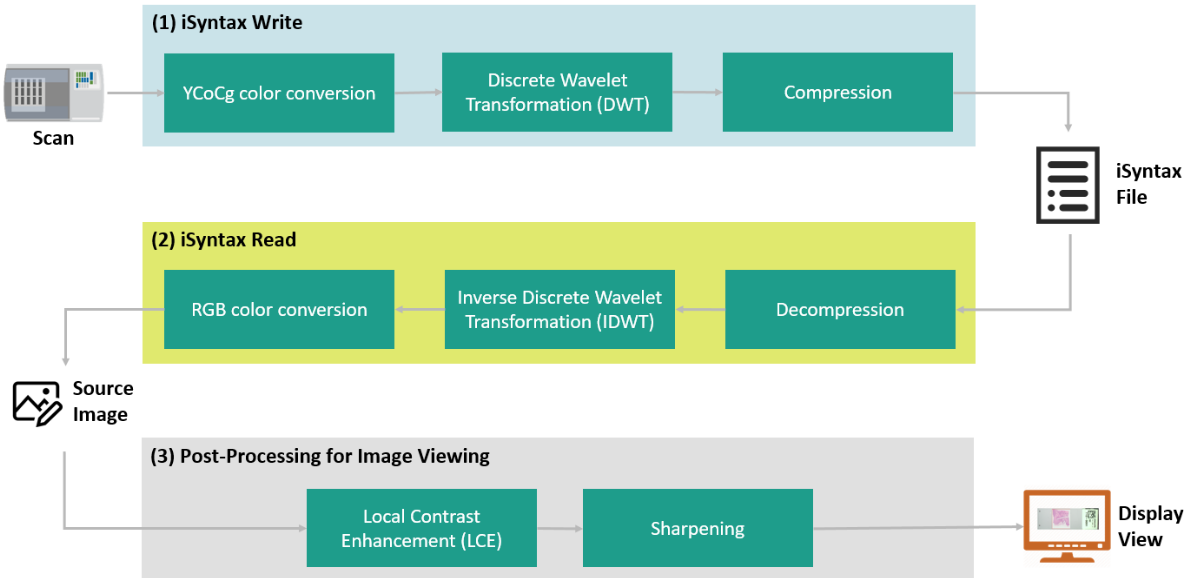

Figure 1 iSyntax image pipeline

4555 207 43941_2020_04_24

About this document

This document describes the file format of iSyntax, i.e. the structure of iSyntax files generated by PIPS.

For more information on the iSyntax image format, refer to the white paper 'Philips iSyntax for Digital Pathology' from Dr. Bas Hulsken, available on the webportal:
www.openpathology.philips.com/index.php/resources/#isyntax

Notice

This document contains source code, which is available as code samples compatible with Python and reference codes compatible with Octave and Matlab. The code samples/reference codes are verified with:

- Python 3.7
- Octave 5.1
- Matlab 9.8

All code samples and reference codes listed in this document are available for download from the Open Pathology Portal at www.openpathology.philips.com.
Please note that the implementation provided in this document is for demonstration purposes and not optimized for maximum performance, nor can it handle very large inputs.

Notice

All the brand and product names are trademarks of their respective companies.

4555 207 43941_2020_04_24

License

2 iSyntax data model

The iSyntax data model represents the whole slide as three images:

- label image, containing slide identification information.
- macro image, providing a thumbnail view of the slide.
- Whole Slide Image (WSI), representing the tissues region of interests, which are scanned at high resolution and stored using the iSyntax compression format.

The data model also contains metadata related to the parameters necessary for image acquisition e.g. scanning protocol, DICOM attributes and acquisition attributes.

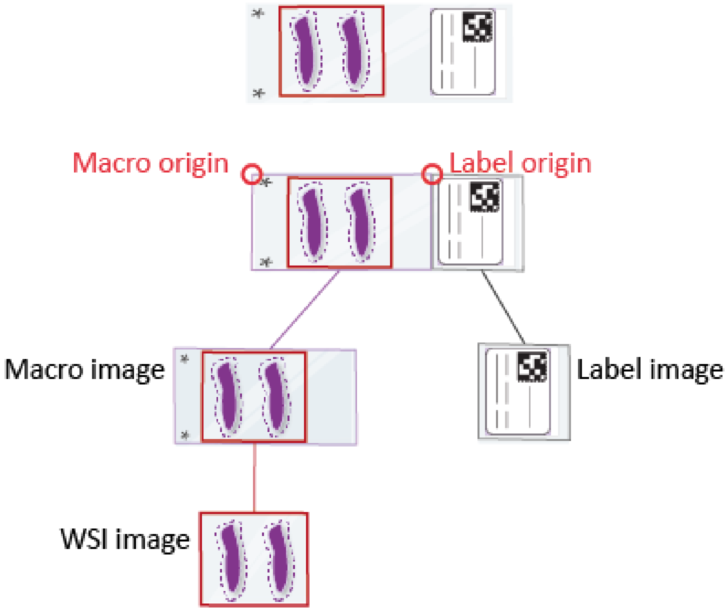

Figure 2 Image representation by sub-images

4555 207 43941_2020_04_24

3 iSyntax file

The iSyntax file is designed to contain both metadata and pixel data corresponding to the iSyntax data model. An iSyntax file is represented by an XML header, End of Table (EOT), optionally a seektable and codeblocks.

XML HEADER	EOT 3 bytes	SEEKTABLE (optional)	CODEBLOCKS

Figure 3 Primary representation of an iSyntax file

XML Header

The XML Header contains the metadata related to the properties describing:

- JPEG image data for the label image, see section Label image
- JPEG image data for the macro image, see section Macro Image
- the WSI

The metadata is stored in UTF-8 encoded XML format.
For more information, see section XML Header.

End of Table (EOT)

The EOT is a marker to indicate that the stream containing the XML Header has ended. EOT represented by 3 characters.

EOT	Hex Representation
“\r \n \x04”	0D 0A 04

Table 1 EOT characters

Seektable

The Seektable is a serialized representation of the block headers as per DICOM standard. It contains the offset and size of the codeblocks.
For more information, see section Seektable structure.

Codeblocks

The recursive Discrete Wavelet Transform (DWT) of the RAW pixel data creates a multiresolution pyramid. Each level in the pyramid is divided into N x N size codeblocks. The codeblocks contain the compressed coefficients. For more information, see section Codeblocks.

Note that the size of the codeblock may vary from scanner to scanner. You can get the size of the codeblock from the XML header in UFS_IMAGE_DIMENSION_RANGE in UFSImageBlockHeaderTemplate dataobject.
For more information, see section Image Dimension Ranges.

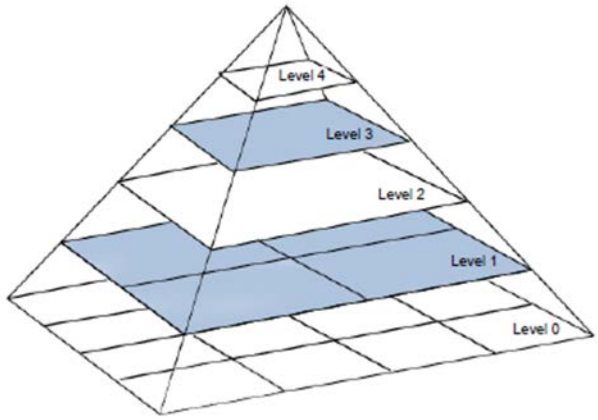

Figure 4 WSI-images pyramid representation

4555 207 43941_2020_04_24

4 XML Header

The XML Header contains the metadata related to the properties describing the WSI and the JPEG image data for both label image and macro image
The metadata is stored in UTF-8 encoded XML format.

The XML Header of the iSyntax file uses three different types of nodes: leaf nodes, branch nodes and array nodes, see Node types.
The root node is a branch node, type ‘DataObject‘ and named ‘DPUfsImport’. For more information, see section DPUfsImport node.

Node types

Leaf node

A leaf node is a node with no child nodes. Generally, a leaf node contains an element named ‘Attribute’. Each leaf node contains four attributes in the same order: Name, Group, Element and PMSVR.

Example of a leaf node:

```
<Attribute Name="DICOM_MANUFACTURER" Group="0x0008" Element="0x0070"
PMSVR="IString">PHILIPS</Attribute>
```

Branch node

A branch node is a node with child nodes, it contains leaf nodes. Generally, a branch node contains an element named ‘DataObject’ and has one attribute: ‘ObjectType’.

Example of a branch node:

```
<DataObject ObjectType="DPScannedImage">
…
…
</DataObject>
```

Array node

Array nodes contains one or more similar type of leaf/branch nodes.

Example of an array node

```
<Attribute Name="UFS_IMAGE_DIMENSION_RANGES" Group="0x301d" Element="0x200a"
PMSVR="IDataObjectArray">
   <Array>
      <DataObject ObjectType="UFSImageDimensionRange">
         <Attribute Name="UFS_IMAGE_DIMENSION_RANGE" Group="0x301d" Element="0x200b"
PMSVR="IUInt32Array">0 1 9215</Attribute>
      </DataObject>
      <DataObject ObjectType="UFSImageDimensionRange">
         <Attribute Name="UFS_IMAGE_DIMENSION_RANGE" Group="0x301d" Element="0x200b"
PMSVR="IUInt32Array">0 1 8191</Attribute>
      </DataObject>
```

4555 207 43941_2020_04_24

```
        <DataObject ObjectType="UFSImageDimensionRange">
           <Attribute Name="UFS_IMAGE_DIMENSION_RANGE" Group="0x301d" Element="0x200b"
PMSVR="IUInt32Array">0 1 2</Attribute>
        </DataObject>
        <DataObject ObjectType="UFSImageDimensionRange">
           <Attribute Name="UFS_IMAGE_DIMENSION_RANGE" Group="0x301d" Element="0x200b"
PMSVR="IUInt32Array">0 1 3</Attribute>
        </DataObject>
        <DataObject ObjectType="UFSImageDimensionRange">
            <Attribute Name="UFS_IMAGE_DIMENSION_RANGE" Group="0x301d" Element="0x200b"
PMSVR="IUInt32Array">0 1 3</Attribute>
        </DataObject>
      </Array>
</Attribute>
```

Metadata attributes

All the attributes with a name starting with ‘DICOM’ are taken from the DICOM standard. For these attributes, the Group and Element form the 4-byte DICOM tag.
All the attributes with a name not starting with ‘DICOM’ are tags which do not exist in the DICOM standard. These are Philips private tags, required for specifying the digital pathology WSI format.

Attributes are composed of:

- Name: the name of the attribute.
- Group: in the format (0xXXXX) in hexadecimal value.
- Element: in the format (0xXXXX) in hexadecimal value,
- PMSVR: describes the data type and format of the attribute value.
- Value: contains the attribute’s data.

Group and Element identify an attribute.

The basic attribute structure is:

```
<Attribute Name="DICOM_MANUFACTURER" Group="0x0008" Element="0x0070"
PMSVR="IString">PHILIPS</Attribute>
```

The following table shows the list of attributes with group tag, element tag and value type.

Attribute Name	Group tag	Element tag	Value type
DICOM_ACQUISITION_DATETIME	0008	002A	IString
DICOM_MANUFACTURER	0008	0070	IString
DICOM_MANUFACTURERS_MODEL_NAME	0008	1090	IString
DICOM_DERIVATION_DESCRIPTION	0008	2111	IString
DICOM_DEVICE_SERIAL_NUMBER	0018	1000	IString
DICOM_SOFTWARE_VERSIONS	0018	1020	IStringArray
DICOM_DATE_OF_LAST_CALIBRATION	0018	1200	IStringArray
DICOM_TIME_OF_LAST_CALIBRATION	0018	1201	IStringArray
DICOM_SAMPLES_PER_PIXEL	0028	0002	IUInt16
DICOM_BITS_ALLOCATED	0028	0100	IUInt16
DICOM_BITS_STORED	0028	0101	IUInt16
DICOM_HIGH_BIT	0028	0102	IUInt16

4555 207 43941_2020_04_24

Attribute Name	Group tag	Element tag	Value type
DICOM_ICCPROFILE	0028	2000	IString
DICOM_LOSSY_IMAGE_COMPRESSION	0028	2110	IString
DICOM_LOSSY_IMAGE_COMPRESSION_RATIO	0028	2112	IDouble
DICOM_LOSSY_IMAGE_COMPRESSION_METHOD	0028	2114	IString
PIIM_DP_SCANNER_RACK_NUMBER	101D	1007	IUInt16
PIIM_DP_SCANNER_SLOT_NUMBER	101D	1008	IUInt16
PIIM_DP_SCANNER_OPERATOR_ID	101D	1009	IString
PIIM_DP_SCANNER_CALIBRATION_STATUS	101D	100A	IString
PIM_DP_UFS_INTERFACE_VERSION	301D	1001	IString
PIM_DP_UFS_BARCODE	301D	1002	IString
PIM_DP_SCANNED_IMAGES	301D	1003	IDataObjectArray
PIM_DP_IMAGE_TYPE	301D	1004	IString
PIM_DP_IMAGE_DATA	301D	1005	IString
PIM_DP_SCANNER_RACK_PRIORITY	301D	1010	IUInt16
DP_COLOR_MANAGEMENT	301D	1013	IDataObjectArray
DP_WAVELET_QUANTIZER_SETTINGS_PER_COLOR	301D	1019	IDataObjectArray
DP_WAVELET_QUANTIZER_SETTINGS_PER_LEVEL	301D	101A	IDataObjectArray
DP_WAVELET_QUANTIZER	301D	101B	IUInt16
DP_WAVELET_DEADZONE	301D	101C	IUInt16
UFS_IMAGE_GENERAL_HEADERS	301D	2000	IDataObjectArray
UFS_IMAGE_NUMBER_OF_BLOCKS	301D	2001	IUInt32
UFS_IMAGE_DIMENSIONS_OVER_BLOCK	301D	2002	IUInt16Array
UFS_IMAGE_DIMENSIONS	301D	2003	IDataObjectArray
UFS_IMAGE_DIMENSION_NAME	301D	2004	IString
UFS_IMAGE_DIMENSION_TYPE	301D	2005	IString
UFS_IMAGE_DIMENSION_UNIT	301D	2006	IString
UFS_IMAGE_DIMENSION_SCALE_FACTOR	301D	2007	IDouble
UFS_IMAGE_DIMENSION_DISCRETE_VALUES_STRING	301D	2008	IStringArray
UFS_IMAGE_BLOCK_HEADER_TEMPLATES	301D	2009	IDataObjectArray
UFS_IMAGE_DIMENSION_RANGES	301D	200A	IDataObjectArray
UFS_IMAGE_DIMENSION_RANGE	301D	200B	IUInt32Array
UFS_IMAGE_DIMENSIONS_IN_BLOCK	301D	200C	IUInt16Array
UFS_IMAGE_BLOCK_HEADERS	301D	200D	IDataObjectArray
UFS_IMAGE_BLOCK_COORDINATE	301D	200E	IUInt32Array
UFS_IMAGE_BLOCK_COMPRESSION_METHOD	301D	200F	IString
UFS_IMAGE_BLOCK_DATA_OFFSET	301D	2010	IUint64
UFS_IMAGE_BLOCK_SIZE	301D	2011	IUint64
UFS_IMAGE_BLOCK_HEADER_TEMPLATE_ID	301D	2012	IUInt32
UFS_IMAGE_BLOCK_HEADER_TABLE	301D	2014	IString

Table 2 List of Attributes

4555 207 43941_2020_04_24

DPUfsImport node

De DPUfsImport node is the root node with the structure:

```
<DataObject ObjectType="DPUfsImport">
…
…
</DataObject>
```

The following table shows the child nodes part of the DPUfsImport.

Parent Data Object: DPUfsImport

Attribute	Group tag	Element tag	Value type	Node type	Description	Range
DICOM_MANUFACTURER	0008	0070	IString dicom:LO	Leaf	DICOM data element (0008,0070) (Value Multiplicity:1)	“PHILIPS”
DICOM_ACQUISITION _DATETIME	0008	002A	IString	Leaf	Date & Time when slide transfer started (XML Header created). DicomAcquisitionDate and DicomAcquisitionTime are combined into one single element	Minimum: 1900-01-01T00:00:00 Maximum: 2154-12-31T23:59:59
DICOM_MANUFACTURERS _MODEL_NAME	0008	1090	IString	Leaf	DICOM data element (0008,1090) (Value Multiplicity:1)	“UFS Scanner”
DICOM_DEVICE_SERIAL _NUMBER	0018	1000	IString	Leaf	DICOM data element (0018,1000) (Value Multiplicity: 1)	“FMTOO19’ Note: Value is configurable in UFS during manufacturing
DICOM_SOFTWARE _VERSIONS	0018	1020	IStringArra y	Leaf	Software versions of two subcomponents. Note: There are no limitations on the number of entries in the list but also no limitations on format/values of the strings in the software versions list.	
DICOM_DATE_OF_LAST _CALIBRATION	0018	1200	IStringArra y	Leaf	Date & Time of last calibration by a service engineer	
DICOM_TIME_OF_LAST _CALIBRATION	0018	1201	IStringArra y	Leaf	Date &Time of last calibration by a service engineer	
PIIM_DP_SCANNER_RACK _NUMBER	101D	1007	IUInt16	Leaf	UFS store position in which the rack was placed and from which the slide was taken.	[1..15]
PIIM_DP_SCANNER_SLOT _NUMBER	101D	1008	IUInt16	Leaf	Position in the rack where the slide was stored.	[1..20]
PIM_DP_UFS_INTERFACE _VERSION	301D	1001	IString dicom:LO	Leaf	Unique identifier of the entire image transfer format	“5.0”
PIM_DP_UFS_BARCODE	301D	1002	IString	Leaf	Base64 encoded Barcode value	N/A

4555 207 43941_2020_04_24

Attribute	Group tag	Element tag	Value type	Node type	Description	Range
PIM_DP_SCANNED_IMAGE S	301D	1003	IDataObje ctArray	Array		N/A
PIIM_DP_SCANNER _OPERATOR_ID	101D	1009	IString	Leaf		"Operator ID"
PIIM_DP_SCANNER _CALIBRATION_STATUS	101D	100A	IString	Leaf	Boolean indicates whether last calibration attempt failed.	"OK" "NOT OK"
PIM_DP_SCANNER_RACK _PRIORITY	301D	1010	IUInt16	Leaf		

Table 3 DPUfsImport node attributes

Scanned Image node

Parent Data Object: DPScannedImage						
Attribute	Group tag	Element tag	Value type	Node type	Description	Range
DICOM_DERIVATION _DESCRIPTION	0008	2111	IString	Leaf	Single string containing image format description	RAW: "Philips UFS V%s" iSyntax: "Philips UFS V%s | Quality=%d | DWT=%d | Compressor=%d" %s= UFS version (RAW+iSyntax) %d: Sucomponent quality (iSyntax only) %d: Transformation Method (iSyntax only) use 1 for legal53 %d: Compression Method (iSyntax only) : use 16 for hulsken
DICOM_LOSSY_IMAGE _COMPRESSION	0028	2110	IString	Leaf	Boolean value: 0 means lossless 1 means lossy	"00" "01"
DICOM_LOSSY_IMAGE _COMPRESSION_RATIO	0028	2112	IDouble	Leaf	Describe lossy image quality/bit reduction Actual compression ratio is unknown, value =1 means lossless: Don't use zero as compression. 1 means no-compression 2 means a factor of 2 compressed.	1 2 3 4
DICOM_LOSSY_IMAGE _COMPRESSION _METHOD	0028	2114	IString	Leaf	Specify custom compression engine. Combination of ImageCompressionMeth od and BlockCompressionMetho d defines exact algorithm.	"PHILIPS_DP_1_0"

4555 207 43941_2020_04_24

Attribute	Group tag	Element tag	Value type	Node type	Description	Range
PIM_DP_IMAGE_TYPE	301D	1004	IString	Leaf	Identifies the image type: Macro, label, WSI	“MACROIMAGE” “LABELIMAGE” “WSI”
PIM_DP_IMAGE_DATA	301D	1005	IString	Leaf	Contains encoded JPEG file of Label Image or Macro Image	NA
DP_COLOR _MANAGEMENT	301D	1013	IDataObjectArray	Array	Specify color management per image.	At most 1 color management object is available per scanned image.
DP_WAVELET _QUANTIZER_SETTINGS_ PER_COLOR	301D	1019	IDataObjectArray	Array	Per color component a list of quantizer settings. First entry belongs to first color component etc… The typical order for colors is either Y-Co-Cg or R-G-B	Typical 2 objects: first for luminance, second for the other color components. Only applicable to WSI.
UFS_IMAGE_GENERAL _HEADERS	301D	2000	IDataObjectArray	Array	General settings regarding the WSI data stream. Image General headers is allowed to contain only one Image General Header.	NA
UFS_IMAGE_BLOCK _HEADER_TEMPLATES	301D	2009	IDataObjectArray	Array	Settings shared by all block headers, can be overridden in individual block headers. Describes properties common to all image blocks(tiles). The common properties are all properties, except the coordinate of the Image block (tile). Image Block Header templates is allowed to contain only one Image Block Header Template.	N=1 Multiple templates may exit: Each ImageBlockHeader might reference to one of these templates. A block header can be fully described by its content.
UFS_IMAGE_BLOCK _HEADERS	301D	200D	IDataObjectArray	Array	Contains one ImageBlockHeader object for each image block in the image. The order of Image Block Header objects in the list corresponds to the order of the image block pixel data in the WSI data stream.	1…n ImageBlockHeaders and ImageBlockheadertable are mutually exclusive; exactly one must be present N must equal to ImageNumberOfBlocks
UFS_IMAGE_BLOCK _HEADER_TABLE	301D	2014	IString	Leaf	Similar to ImageBlockHeaders except that each ImageBlockHeader is in binary format: the entire table is base64 encoded.	ImageBlockHeaders and ImageBlockHeaderTable are mutually exclusive; exactly one must be present. Number of ImageBlockHeaders must equal to ImageNumberOfBlocks

Table 4 DPScannedImage node attributes

Image General Header

Parent Data Object: UFSImageGeneralHeader						
Attribute	**Group tag**	**Element tag**	**Value type**	**Node type**	**Description**	**Range**
UFS_IMAGE_NUMBER_OF_BLOCKS	301D	2001	IUInt32	Leaf	Total number of datablocks (tiles) in WSI data stream.	0.. 2^32-1
UFS_IMAGE_DIMENSIONS_OVER_BLOCK	301D	2002	IUInt16Array	Leaf	Defines interpretation of ImageBlockCoordinate values; it tells for each coordinate into which dimension it maps	0..4 RAW: Fixed value: [0 1 2] meaning [x y color] iSyntax: Fixed value [0 1 2 3 4] meaning [x y color scale waveletcoef]
UFS_IMAGE_DIMENSIONS	301D	2003	IDataObjectArray	Array	List of all dimensions in WSI data stream. Order of dimensions is fixed by order in this list. Dimension 0 is the sequence number of the first dimension in the list.	RAW: Only “x” “y” “component” iSyntax: Also include “scale” and “waveletcoef” (in this order)
UFS_IMAGE_DIMENSION_RANGES	301D	200A	IDataObjectArray	Array		Specify the values for the entire image

Table 5 UFSImageGeneralHeader node attributes

4555 207 43941_2020_04_24

Image Dimensions

Parent Data Object: UFSImageDimension						
Attribute	**Group tag**	**Element tag**	**Value type**	**Node type**	**Description**	**Range**
UFS_IMAGE_DIMENSION_NAME	301D	2004	IString	Leaf	Name of dimension.	“x” “y” “component” “scale” “waveletcoef” X = short side of slide Y = long side of slide (see appendix A) Component refers to color Scale=” length of coedfficient” (use subcomponent dwt_coefficient) Waveletcoef = spatial frequency component (use subcomponent coefficient_type)

Attribute	Group tag	Element tag	Value type	Node type	Description	Range
UFS_IMAGE_DIMENSION_TYPE	301D	2005	IString	Leaf	Physical characteristics of dimension	“spatial” “colour component” “scale” “waveletcoef” Spatial: continuous dimension Colour: discrete named dimension Scale: discrete unnamed dimension waveletcoef: discrete named dimension
UFS_IMAGE_DIMENSION_UNIT	301D	2006	IString	Leaf	Unit of dimension (S.I. name)	“CentiMeter” “MilliMeter” “MicroMeter” “NanoMeter” “PicoMeter” Only “MicroMeter” is used
UFS_IMAGE_DIMENSION_SCALE_FACTOR	301D	2007	IDouble	Leaf	Physical increment of dimension	Greater than 0 Scientific format allowed (Like: “250.0E-03”
UFS_IMAGE_DIMENSION_DISCRETE_VALUES_STRING	301D	2008	IStringArray	Leaf	Names of discrete dimension values	“R” “G” “B” “Y” “Co” “Cg” “LL” “LH” “HL” “HH” RAW colors: Fixed value [“R” “G” “B”]. iSyntax lossless colors: Fixed value [“R” “G” “B”] iSyntax lossy colors: Fixed value [“Y” “Co” “Cg”] iSyntax waveletcoefs: Fixed value [“LL” “LH” “HL” “HH”]

Table 6 UFSImageDimension node attribute

4555 207 43941_2020_04_24

Image Dimension Ranges

Parent Data Object: UFSImageDimensionRange

Attribute	Group tag	Element tag	Value type	Node type	Description	Range
UFS_IMAGE_DIMENSION_RANGE	301D	200B	IUInt32Array	Leaf	Define a range by specifying 3 values: - start value - step value - end value Applicable for attributes within the UFSImageGeneralheader dataobject	UFSImageGeneralHeader: calculate ranges for entire image: Imagesize = MX x MY Imagetopleft = (NX, NY) Scale = max_dwt_level(0-based) Max_dwt_level = (nrOfDwtLevels-1) For RAW 3 entries: x coordinate: [NX 1 NX+MX-1] y coordinate: [NY 1 NY+MY-1] color coordinate: [0 1 2] for iSyntax 5 entries: x coordinate: [NX 1 NX+MX-1] y coordinate: [NY 1 NY+MY-1] color coordinate: RGB use [0 1 2] color coordinate: YCoCg use [0 1 2] scale coordinate: [0 1 max_dwt_level] waveletcoef coordinate: use [0 1 3
UFS_IMAGE_DIMENSION_RANGE	301D	200B	IUInt32Array	Leaf	Define a range by specifying 3 values: - start value - step value - end value Applicable for attributes within the UFSImageBlockHeaderTemplate dataobject	UFSImageBlockHeaderTemplate: specify ranges for block template For RAW 3 entries: x coordinate: [0 1 1023] y coordinate: [0 1 1023] {blocksize 1024x1024} color coordinate: [0 1 2] for iSyntax 5 entries: x coordinate: [0 1 127] * (2^dwt_level) y coordinate: [0 1 127] * (2^dwt_level) color coordinate: R use [0 0 0], G use [1 0 1], B use [2 0 2] color coordinate: Y use [0 0 0], Co use [1 0 1], Cg [2 0 2] scale coordinate: [dwt_level 0 dwt_level] waveletcoef coordinate: LL use [0 0 0], else use [1 1 3]

Table 7 UFSImageDimension node attribute

Block Header Templates

Parent Data Object: UFSImageBlockHeaderTemplate						
Attribute	Group tag	Element tag	Value type	Node type	Description	Range
DICOM_SAMPLES_PER_ PIXEL	0028	0002	IUInt16	Leaf	The number of sample within each pixel	1… n Fixed value: 3
DICOM_BITS_ALLOCATED	0028	0100	IUInt16	Leaf	The number of bits that one sample occupies.	1… n RAW: Fixed value 8 iSyntax: Fixed value 16
DICOM_BITS_STORED	0028	0101	IUInt16	Leaf	The number of used bits of one sample within the occupied area	1 ... DicomBitsAllocated RAW: Fixed value 8 iSyntax: Fixed value 16
DICOM_HIGH_BIT	0028	0102	IUInt16	Leaf	The position of the MSB of each sample within the occupied area	(DicomBitsStored-1) ...(DicomBitsAllocated-1) RAW: Fixed value 7 iSyntax: fixed value 15
DICOM_PIXEL_ REPRESENTATION	0028	0103	IUInt16	Leaf	1 indicating signed pixels (2's complement) 0 indicating un-signed pixel values	0: unsigned bytes 1: signed bytes
UFS_IMAGE_DIMENSION_ RANGES	301D	200A	IDataObjectArray	Array	Contains an Image Dimension range for each dimension in the block. Length(ImageDimensionRanges) = length(ImageDimensions)	Specify the values for all blocks that inherit from this BlockHeaderTemplate
UFS_IMAGE_DIMENSION_ IN_BLOCK	301D	200C	IUInt16Array	Leaf	Defines mapping from linear sample space to multidimensional space: index of first iterating dimension, …, until index of last iterating dimension	RAW: Fixed value: [2 1 0] meaning [color y x] iSyntax: Fixed value:[1 0 4] meaning [y x waveletcoef]
UFS_IMAGE_BLOCK_ COMPRESSION_METHOD	301D	200F	IString	Leaf	Specify used method of compression	Numeric integer value 16 for hulsken compression 19 for hulsken2 compression

Table 8 UFSImageBlockHeaderTemplate node attribute

4555 207 43941_2020_04_24

Block Headers

Parent Data Object: UFSImageBlockHeader						
Attribute	Group tag	Element tag	Value type	Node type	Description	Range
UFS_IMAGE_BLOCK_ COORDINATE	301D	200E	IUInt32 Array	Leaf	Position of the image block(tile) along the dimensions specified in the Image General Header – Image Dimensions Over Block,	x: 0... (108000-1024-1) y: 0... (308000-1024-1) component: 0 1 2 scale: 0 1...(ndwtLevels-1) waveletcoef: 0...3 [x y] (in that order, see DimensionOverBlock) Note: max image size is 100000x240000 (25x60 mm) but max slide size is 27x77 mm, and origin is defined at corner of slide (and not at corner of scanned image) For RAW, component is always 0
UFS_IMAGE_BLOCK_ TEMPLATE_ID	301D	2012	IUInt32	Leaf	Index of template, given in element Image Block Header Template	0… (NrOfTemplates-1) 0-Based index

Table 9 UFSImageBlockHeader node attribute

4555 207 43941_2020_04_24

Image Color Management

Parent Data Object: DPColorManagement						
Attribute	Group tag	Element tag	Value type	Node type	Description	Range
DICOM_ICCPROFILE	0028	2000	IString	Leaf	Base64 encoded ICC profile	NA

Table 10 DPColorManagement node attribute

Wavelet Quantizer Setting

Parent Data Object: DPWaveletQuantizerSettingsPerColor

Attribute	Group tag	Element tag	Value type	Node type	Description	Range
DP_WAVELET_ QUANTIZER_SETTINGS_ PER_LEVEL	301D	101A	IDataObjectArray	Array	Per level a list of quantizer settings. First settings belong to level 0, second to level 1 etc..	NA

Table 11 DPWaveletQuantizerSettingsPerColor node attribute

Wavelet Quantizer Setting Per Level

Parent Data Object: DPWaveletQuantizerSettingsPerLevel

Attribute	Group tag	Element tag	Value type	Node type	Description	Range
DP_WAVELET_ QUANTIZER	301D	101B	IUInt16	Leaf	Wavelet coefficients are rounded to 2^quantizer.	0 ... n
DP_WAVELET_ DEADZONE	301D	101C	IUInt16	Leaf	Wavelet coefficients are pulled to zero for absolute values greater or equal to dead zone.	0 … n

Table 12 DPWaveletQuantizerSettingsPerLevel node attribute

5 Codeblocks

The codeblocks section in the iSyntax file contains the compressed pixel data for the WSI. This data is generated by a recursive DWT of the original image to create the multiresolution pyramid.

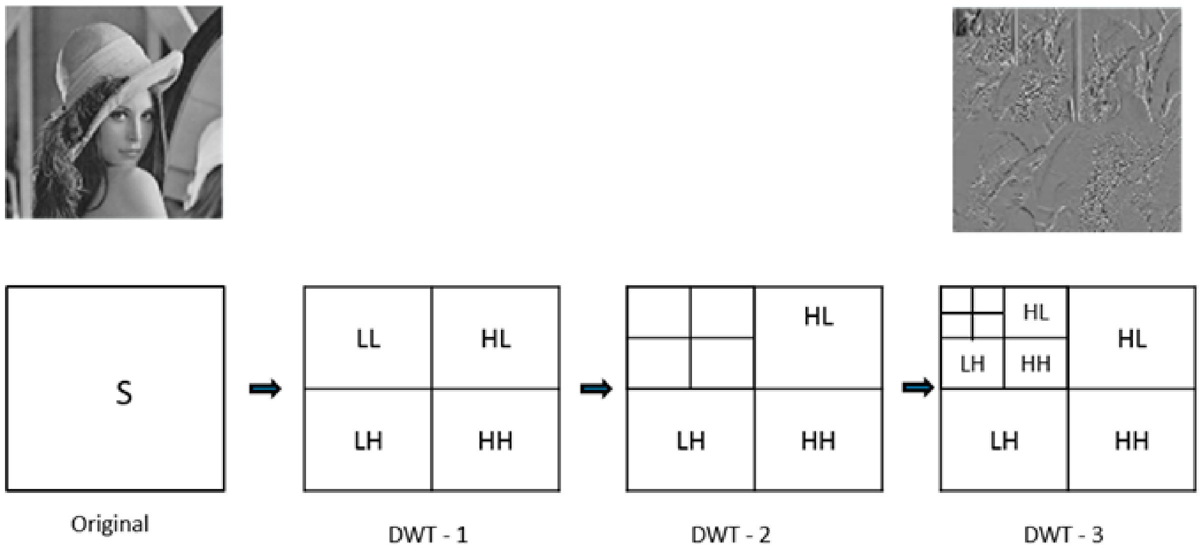

Figure 5 Three level recursive DWT

The transformed coefficients HL, LH and HH are spatially aligned and merged together to one block [HL, LH, HH]. After merging, the LL and [HL, LH, HH] are divided into codeblocks of the respective color channels. Each codeblock is compressed using the Hulsken compression method.

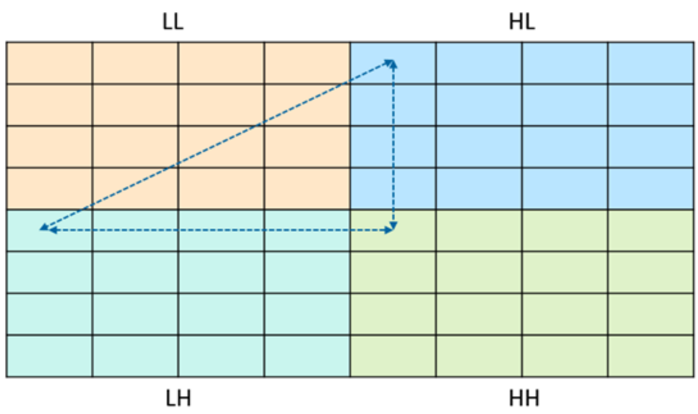

Figure 6 Spatially aligned coefficient blocks (HL, LH, HH)

All the attributes describing the WSI are part of the data object 'ScannedImage'. For more information see the table in section Scanned Image node where image type (PIM_DP_IMAGE_TYPE) will be 'WSI'.

4555 207 43941_2020_04_24

Codeblock structure

Each codeblock starts with a DICOM sequence tag (0xFF FE, 0xE0 00) followed by data size of 4-bytes. The codeblock contains the compressed coefficients of the WSI. All codeblocks data varies in size.

All values are stored in little-endian representation.

0xFFFE, 0xE000	SIZE (4 BYTES)	CODEBLOCK

Figure 7 Codeblock structure

Codeblock packaging scheme

An iSyntax file is composed of

- *XML header*: metadata related to the properties describing the WSI and other properties.
- *EOT*: 3 characters' marker to delimit the XML Header and pixel data part.
- *Start of pixel data*: DICOM tag (0x7FE0, 0x0010) followed by a 4-byte length of pixel data. Usually this will be filled with (0xFFFF, 0xFFFF) which means unknown length. It's the start of the pixel data.
- *Seektable*: DICOM serialized HeaderBlocks (optional).
- *Codeblocks*: the compressed coefficients.

All values are stored in little-endian representation

<table>
<tr><td colspan="5">XML Header</td></tr>
<tr><td colspan="5">EOT (3 bytes)</td></tr>
<tr><td colspan="5">0x7FE0, 0x0010, 0xFFFF, 0xFFFF (Optional)</td></tr>
<tr><td>0x301D, 0x2015</td><td>SIZE (4 BYTES)</td><td>SEEKTABLE (Optional)</td><td>0xFFFE, 0xEODD</td><td>0x0000, 0x0000</td></tr>
<tr><td>0xFFFE, 0xE000</td><td>SIZE (4 BYTES)</td><td colspan="3">CODEBLOCK</td></tr>
<tr><td>0xFFFE, 0xE000</td><td>SIZE (4 BYTES)</td><td colspan="3">CODEBLOCK</td></tr>
<tr><td colspan="5">…</td></tr>
<tr><td>0xFFFE, 0xE000</td><td>SIZE (4 BYTES)</td><td colspan="3">CODEBLOCK</td></tr>
</table>

Figure 8 iSyntax file detailed representation

4555 207 43941_2020_04_24

After the DWT, the coefficients data is divided into 128x128 size codeblocks.
These codeblocks are packaged in the iSyntax files in a specific order.

For each block in LL
- Per Channel : Y, Co, Cg
  - Per Level : NumLevels to 0 (increasing resolution)
    - Per Spatially aligned block: Block 1, ..., N
      - HL, LH, HH: coefficients
  - LL block

Figure 9 Packaging order

The codeblocks lay on a grid, which can span a significantly larger area than the area that is scanned by the UFS. This grid is a rounded multiple of the block dimensions of the top image in the base image coordinates. Every grid of a higher level is reduced by factor of 2. The codeblocks are written from left to right, from top to bottom, from top level to base level and finally the first to last color channel.

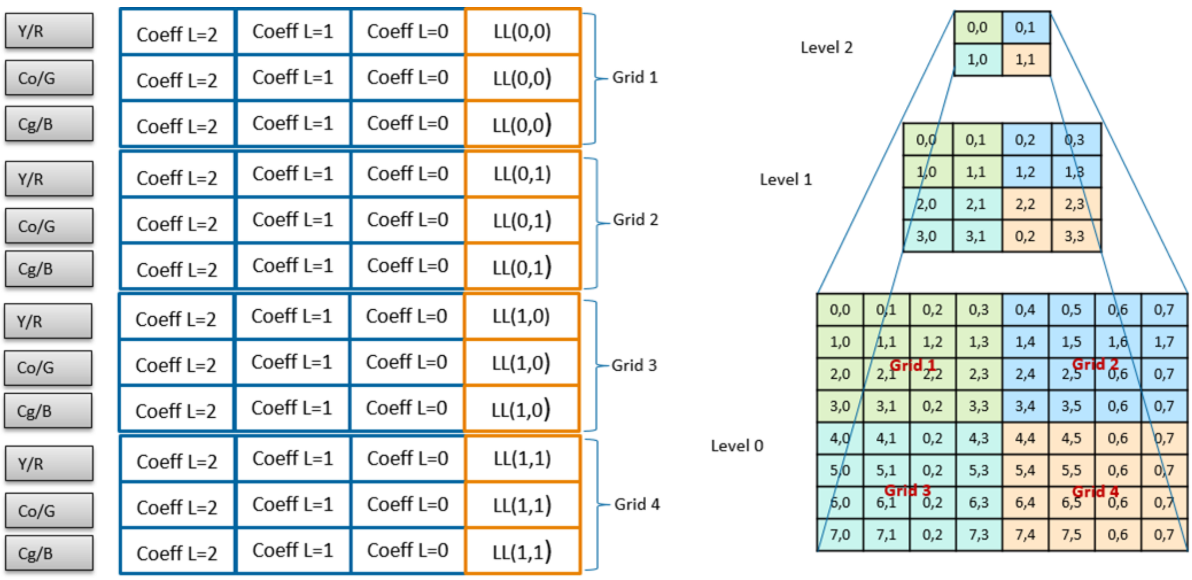

Figure 10 Packaging structure

4555 207 43941_2020_04_24

Image block header structure

The attribute UFS_IMAGE_BLOCK_HEADER_TABLE (301D, 2014) for WSI's, contain the base64 encoded DICOM serialized block header values. This attribute contains the information of all block headers.
Each codeblock contains a block header, which is required to locate the codeblock in a file.
A block header is composed of:

- *Block Coordinates*: composed of x coordinate, y coordinate, color channel, scale (dwt level), Coefficient (0 for LL, 1 for HL, LH, HH).
- *Block data offset*: file offset for codeblock.
- *Block Size*: size of the codeblock.
- *Block Header Template Id*: the properties common to all codeblocks (image blocks). See the table in section Block Header Templates.

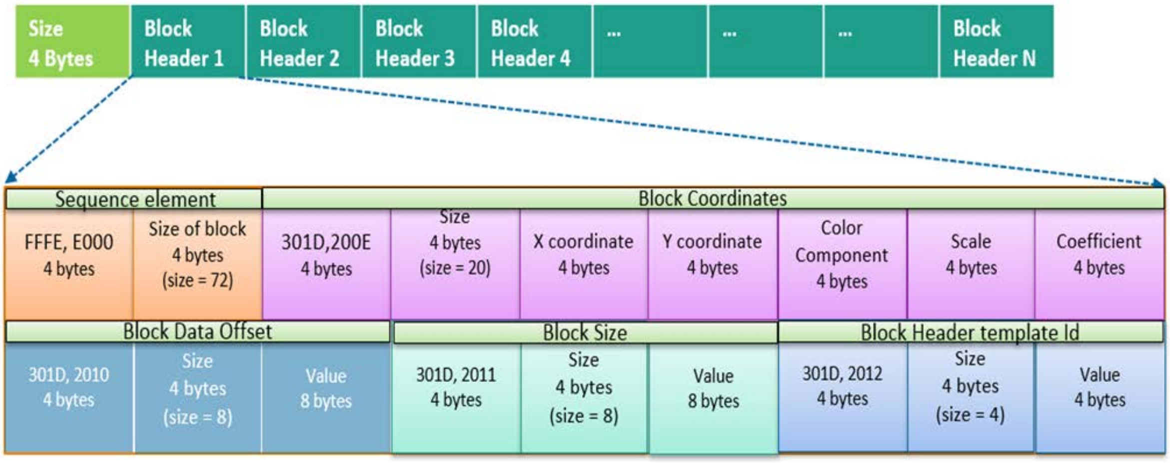

Figure 11 Image block header structure

4555 207 43941_2020_04_24

The images generated by the UFS do not have full information in the image block header table. i.e. Block Data Offset and Block Size information will not present. This information is stored as part of the seektable.
It is essential to map this information by knowing the order of the codeblocks in the seektable. See section Seektable structure.

Codeblocks representation for WSI

WSI's can have multiple scanned tissue regions as shown by the red color rectangles in the figure below. A grid of codeblocks represent the WSI. The coordinates for the codeblocks that are outside the scanned tissue boundaries (shown in a yellow color in the figure below) are set to zero in the block header table. These codeblocks are not written in the iSyntax file.

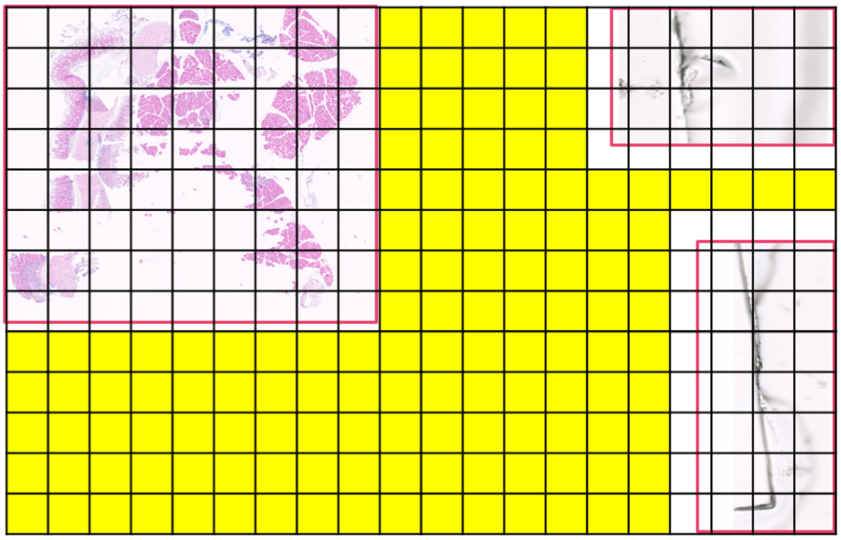

Figure 12 Image block header structure

4555 207 43941_2020_04_24

6 Seektable structure

Seektables refer to the serialized representation of headerblocks as per the DICOM standard. Headerblocks contain the offset and size of the codeblocks. The seektable is an optional part in the iSyntax file.
If the seektable is present, the image block header information is partly stored as part of the attribute UFS_IMAGE_BLOCK_HEADER_TABLE (301D, 2014) and partly by the seektable. It is essential to map the block header information and seektable information to read the codeblocks. For more information, see section Image block header structure.

- Seektables start with a DICOM tag 0x301D, 0x2015.
- Seektables are composed of codeblock headers. Each codeblock header is composed of the ImageBlockDataOffset (0x301D, 0x2010) and ImageBlockSize (0x301D, 0x2011).

Description	Start of seektable			Codeblock 0											Codeblock 1	...	Codeblock N-1	Endoftable		
				Start			ImageBlockDataOffset				ImageBlockSize									
Name	group	element	size	group	element	size	group	element	size	data	group	element	size	data				group	element	size
Size [bytes]	2	2	4	2	2	4	2	2	4	8	2	2	4	8	...	...	...	2	2	4
Value	0x301D	0x2015	0xFFFFFFFF	0xFFFE	0xE000	32	0x301D	0x2010	8	<offset>	0x301D	0x2011	8	<size>	...	...	...	0xFFFE	0XE0DD	0

Figure 13 Seektable structure

Zero padding

In order to map the spatial arrangement of codeblocks with respect to the WSI, it is useful to understand the concept of zero padding. Zero padding is applied to the input image to create a WSI.

At first, the image is padded uniformly on all sides. The amount of padding depends on the number of DWT levels. It is calculated as per the equation below.
The wavelet (PerLevelPadding) and the number of DWT-levels determine the number of padded black pixels. Note that for the wavelet transformation Legall5/3 the PerLevelPadding is 3.

$$Padding = (PerLevelPadding << NrLevels) - PerLevelPadding$$

Equation 1 Codeblock grid dimensions

Secondly, the image is padded further on the right and bottom side. This to ensure that the dimensions of the image are a multiple of the codeblock size. The base level image needs to be sufficiently padded to make sure that the image dimensions for the highest DWT level are also a multiple of the codeblock size
The codeblock grid dimensions (gridWidth x gridHeight) are a function of the image dimensions (width x height), the codeblock dimensions (BlockWidth x BlockHeight) and the number of DWT-levels performed.
The width and height are specified in the XML Header.

$$gridWidth = \left(\frac{(width + (BlockWidth << NrLevels) - 1)}{BlockWidth << NrLevels} \right) << (NrLevels - 1)$$

$$gridHeight = \left(\frac{(height + (BlockHeight << NrLevels) - 1)}{BlockHeight << NrLevels} \right) << (NrLevels - 1)$$

Equation 2 Codeblock grid dimensions

4555 207 43941_2020_04_24

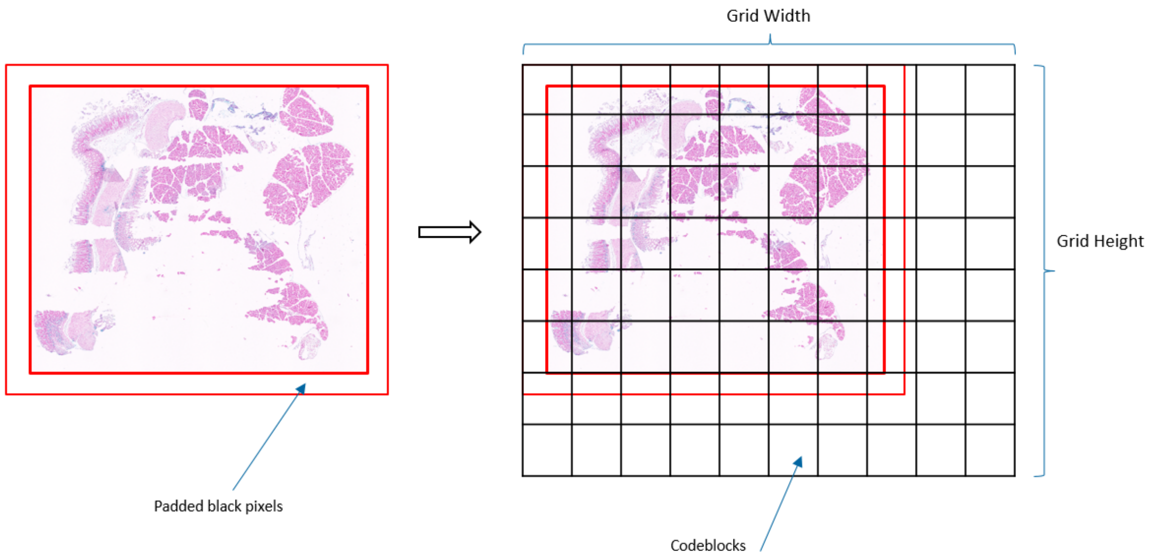

Figure 14 Padding and codeblock grid

The number of codeblocks that are encoded in the seektable can be significantly greater than the actually stored codeblocks. The actual number of codeblocks is determined by the area that is scanned by the scanner.
Every grid of a higher level is reduced by a factor 2. This grid also determines the IDs of the codeblocks, which are the unique numbers that define the position of a codeblock. This codeblock ID is counted from left to right, from top to bottom, from the base level to the top level and finally from the first to the last color channel of the codeblock grid.

7 Reading macro images and label images

The macro image and the label image are a two-dimensional image. These images can be traversed over the X and Y range (width and height respectively) with an incremental step size of 1 to access pixel data.

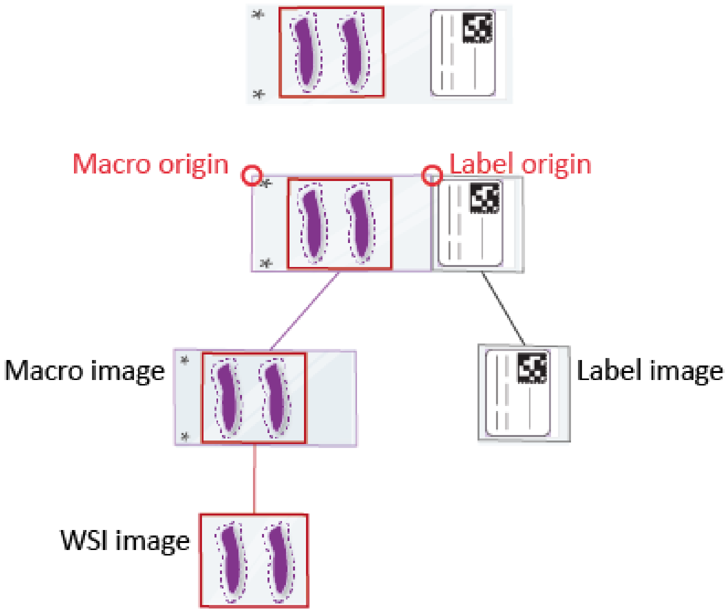

Figure 15 Image representation by sub-images

Label image

The label image contains slide identification information. The label image is a JPEG image encoded with base64 encoding.
All the attributes related to the label image are part of the data object ‘ScannedImage’. Refer to the table in section Scanned Image node.

- The image type (PIM_DP_IMAGE_TYPE) will be ‘LABLEIMAGE’.
- The image data is part of the attribute ‘PIM_DP_IMAGE_DATA’.

```
<DataObject ObjectType="DPScannedImage">
        <Attribute Name="PIM_DP_IMAGE_DATA" Group="0x301D" Element="0x1005"
PMSVR="IString">
        ...
        </Attribute>
        <Attribute Name="PIM_DP_IMAGE_TYPE" Group="0x301D" Element="0x1004"
PMSVR="IString">LABELIMAGE</Attribute>
        ...
</DataObject>
```

NOTICE
The Python code sample ‘Extract label image’ demonstrates to extract the label image from an iSyntax file.

```
$python extract_macro_label_image.py “<iSyntax file path>”
```


4555 207 43941_2020_04_24

Macro Image

The macro image provides a thumbnail view of the slide. The macro image is a JPEG image encoded with base64 encoding.
All the attributes related to macro image are part of the data object 'ScannedImage'. Refer to the table in section Scanned Image node.

- The mage type (PIM_DP_IMAGE_TYPE) will be 'MACROIMAGE'
- The image data is part of the attribute 'PIM_DP_IMAGE_DATA'.

```
<DataObject ObjectType="DPScannedImage">
       <Attribute Name="PIM_DP_IMAGE_DATA" Group="0x301D" Element="0x1005"
PMSVR="IString">
       ...
       </Attribute>
       <Attribute Name="PIM_DP_IMAGE_TYPE" Group="0x301D" Element="0x1004"
PMSVR="IString">MACROIMAGE</Attribute>
       ...
</DataObject>
```

NOTICE

The Python code sample 'Extract macro image' demonstrates to extract the macro image from an iSyntax file.

```
$python extract_macro_label_image.py "<iSyntax file path>"
```

4555 207 43941_2020_04_24

8 Reading WSI images

The compressed pixel data are stored as codeblocks in the iSyntax file. This section demonstrates the following with the help of sample codes:

1. Extraction of codeblocks from an iSyntax file, for more information see section [Extract codeblocks from the iSyntax file](#).
2. Reconstruction of original image from the codeblocks, for more information see section [Reconstruct the original image](#).

The lowest resolution image is reconstructed using LL coefficient codeblocks. To reconstruct the image at the higher resolution levels, a recursive inverse DWT is required.

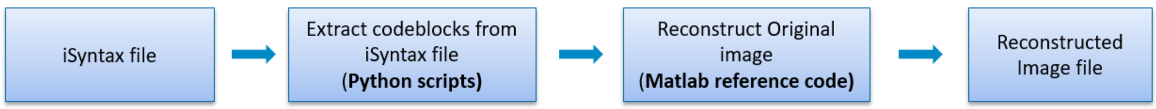

Figure 16 Reconstructing WSI image from an iSyntax file

The following image shows an iSyntax file containing 2 levels of inverse DWT.

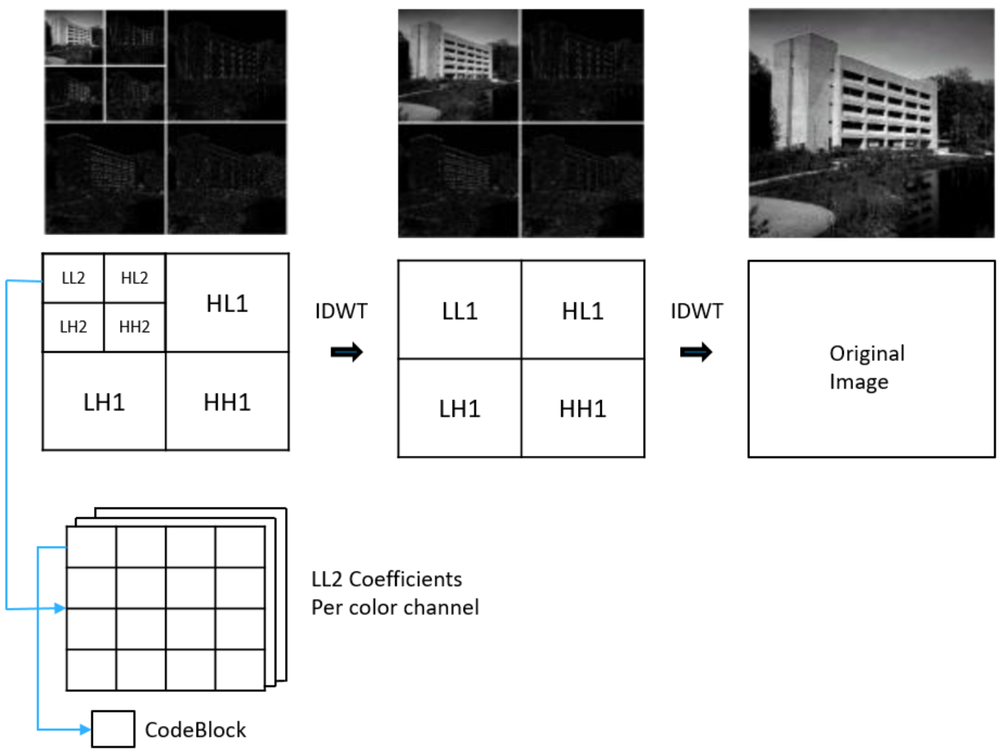

Figure 17 Two-level inverse DWT

Extract codeblocks from the iSyntax file

Codeblocks can be extracted from the iSyntax file, see the Python code sample in the NOTICE.
The transformed coefficients (LL2, [HL2 LH2, HH2] and [HL1, LH1, HH1]) are divided into a number of codeblocks. The number of codeblocks depend on the width and height of the image. In this example the size of each codeblock is 128x128.

The size of each codeblock may vary from scanner to scanner. The size of the codeblock is available in the XML header in the metadata attribute 'UFS_IMAGE_DIMENSION_RANGE', data object 'UFSImageBlockHeaderTemplate', see section Image Dimension Ranges.

NOTICE

The Python code sample 'Extract codeblocks' demonstrates extractions all codeblocks from an iSyntax file.

```
$python extract_codeblocks.py level -p
```

In this code sample:

- 'level' is an integer value that corresponds to the DWT level. You can pass '-1' to extract all the level's codeblocks.
- the script internally de-serializes the Block Header Table, see section Image block header structure. Codeblocks are written in different .ssv files with format: "Codeblock_{x_coordinate}_{y_coordinate}_{color_component}_{scale}_{block_header_template}.ssv"
- the script reads the codeblocks using the block data offset and block size. The codeblocks are written in .ssv files.
- option '–p' generates the properties .csv file containing a number of DWT levels, codeblock dimensions and width and height of the WSI image. These properties are required in Matlab to reconstruct the original WSI image.

4555 207 43941_2020_04_24

Reconstruct the original image

Pixel data can be reconstructed from the codeblocks according the specifications of the image compression format described in the 'Pathology iSyntax Compression' from Dr. Bas Hulsken, available on the webportal:
www.openpathology.philips.com/index.php/resources/#isyntax

To reconstruct the original image:

- decompress the codeblocks using the Hulsken decompression method
- identify the spatially related codeblocks by using the image block header structure (x coordinate, y coordinate, color channel, scale (DWT level) and coefficient), see section Image block header structure)
- stitch the spatially related codeblocks together to create the coefficients of the respective color channels
- perform a two-level inverse DWT
- post-processing: transforms the YCoCg colorspace to an RGB colorspace (the color space of the WSI image is stored in XML header, see section Image Dimensions)

In the following example, an inverse DWT is performed for the LL2, HL2, LH2, HH2 coefficients to create LL1. An inverse DWT is then performed for this generated LL1 along with [HL1, LH1, HH1] to create the original WSI.

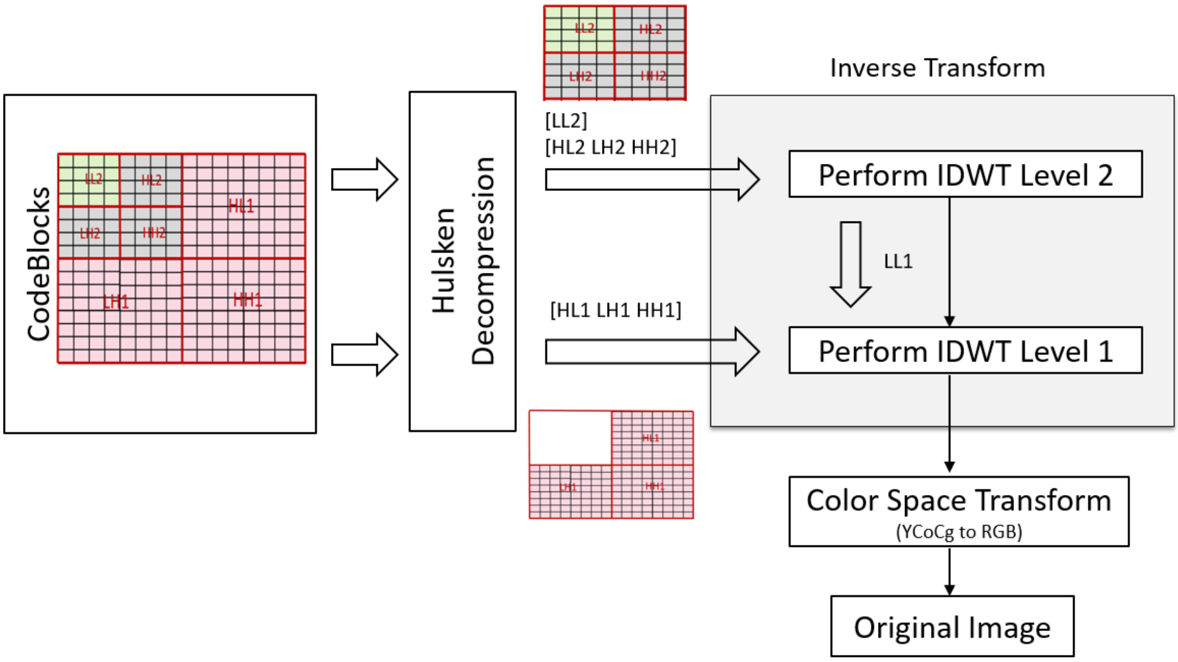

Figure 18 Two-level inverse DWT

NOTICE

The following function in Matlab reference code demonstrates the decompression of the codeblocks.

```
decompressed_block = hulskendecompress(double(load(file)), 16, [128, 128], 1)
```

The function parameters are:

- compressed codeblock file path
- number of bits, 16
- codeblock dimension (this value can vary), [128, 128]
- hulsken compression version, 1

The following function in Matlab reference code demonstrates the inverse DWT.

```
reconstructed_LL = wavelet2dilift(LL, HL, LH, HH, LS, width, height)
```

The function parameters are:

- coefficients, LL, HL, LH, HH
- lifting scheme (LS), iSyntax uses rbio2.2 (legall5/3)
- width
- height

The following function in Matlab reference code demonstrates post-processing: transforms YCoCg color space to RGB color space

```
image_out = ycocg2rgb(image_in)
```

The function parameters are:

- pass image

The following function in Matlab reference code demonstrates post-processing: transforms RGB color space to YCoCg color space

```
image_out = rgb2ycocg(image_in)
```

The function parameters are:

- pass image

The following function in Matlab reference code demonstrates decompressing of all the codeblocks to the original image in one go. This reference code performs the decompression, then stiches the codeblocks to generate the coefficients level wise and color channel wise, and then performs IDWT to generate the original WSI.

```
reconstructed_image = reconstruct_image(<foldername>)
```

The function parameters are:

- folder name, containing all the codeblocks extracted from python sample along with the properties of the iSyntax file

4555 207 43941_2020_04_24

Appendix

Coordinate system

The WSI and the macro image have the same orientation.

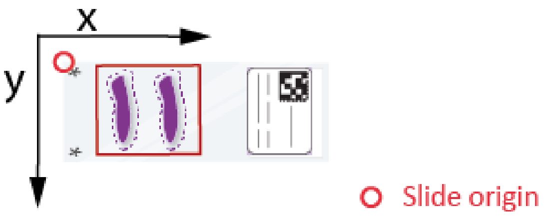

Figure 19 Coordinate system

4555 207 43941_2020_04_24

Philips

Philips Medical Systems Nederland B.V.
Veenpluis 6
5684 PC Best
The Netherlands

www.openpathology.philips.com

Printed in the Netherlands
4522 207 43941 * 2020-APR-24 en