SZI format

Overview and Description of file structure

www.pathozoom.com

www.smartinmedia.com

Martin Weihrauch, M. D.

Elsternweg 6

50997 Köln

Germany

V1.1 – 27.02.2018

1. Background / History

Using the Microsoft Deep Zoom format to serve tiles from high-resolution images shows a good performance for panning and zooming with JavaScript programs like OpenLayers or OpenSeaDragon in browsers. The tiles lie in folders of each zoom level directly on the hard drive and can be quickly accessed by the server.

However, if tiled images need to be moved from one server to another, the copy speed is very low. For instance, if there are 1,000 images of an average resolution of 100,000 x 100,000, approximately 100 million tiles are stored on the hard drive. Moving 100 million files (instead of 1,000 files) is extremely slow and impracticable. Thus, a single-file format is desirable for serving high-resolution images.

John Cupitt (Imperial College London) and Martin Weihrauch (Smart In Media, Germany) created the concept of a single-file format for high-resolution images and named it “Smart Zoom Image format” or SZI format. In the SZI file format, the Microsoft deep zoom pyramid (folders) is stored in an uncompressed ZIP file.

2. Concept of the SZI format

High-resolution images, which are served in tiled pieces, can have different tile layouts. The most used layouts are the “Microsoft Deep Zoom” format, the “Zoomify” format and the “Google Maps” format. For SZI, the Microsoft Deep Zoom format was chosen (although the szi-server.php by Martin Weihrauch can also handle Zoomify layout) and will be described below.

However, the high-resolution images on a server should be well „transportable“ in order to, for example, copy them from one server to another or to use them for image analysis, etc. The best way to reduce the problems with handling millions of small tiles on server hard drives is putting these tiles in a container file.

In order to not „reinvent the wheel“ of container formats, John Cupitt and Martin Weihrauch decided to choose the well-known ZIP-format as the container.

In fact, the SZI format is not a newly invented format, it is just a combination of the techniques Microsoft Deep Zoom tile layout and a ZIP container. However, for an easier communication, the term “format” will still be used.

Therefore, the ZIP-file will be renamed to *.szi. The reason for that is that future users will not be confused, whether a file is a regular ZIP archive on their hard drive or a Smart Zoom image file. We will still be able to open the files with ZIP, we just have to associate our ZIP-programs like WinRAR, etc with the “*.szi” format.

For serving the tiles, the server program should not – for each tile request – read the ZIP with a regular ZIP library, but write a map file once before the first served tile. Reading an entire ZIP directory for each tile, which has to be served (of a ZIP file with possibly millions of contained tiles) is too inefficient. Creating a map file, considerably speeds up the process. The design of such a map, the Smart Zoom Directory file (SZD) will be discussed later and is not important for just outputting SZI files, e. g. from a scanner.

3. The Microsoft Deep Zoom format

3.1.Overview

First, this is an explanation by Microsoft

(https://msdn.microsoft.com/en-us/library/cc645077(v=vs.95).aspx)

"*Single large images in Deep Zoom are represented by a tiled image pyramid. This allows the Deep Zoom rendering engine to grab only that bit of data that is necessary for a particular view of an image. If an image is being viewed zoomed out very far, then a small thumbnail is all that's needed to show the image on screen.*

However, if the user is zoomed in to a specific area of a large image, then only those tiles needed to show the specific areas are downloaded. This can lead to very large bandwidth savings because often only some aspects of a large image are interesting to the user. The illustration below shows what the image pyramid looks like conceptually. An image is stored as a tiled image pyramid. At each level of the pyramid, the image is scaled down by 4 (a factor of 2 in each dimension). Also, the image is tiled up into 256x256 tiles.

If, for example, you were zoomed in to see only the highlighted middle part of the image, Deep Zoom only loads the highlighted tiles, instead of the entire 1024x1024 image.

Each resolution of the pyramid is called a level. Levels are counted from the 1x1 pixel as level 0. Each level is the size 2(level)x2(level). Each level is stored in a separate folder. All levels are stored in a folder with the same name as the DZI file with the extension removed and "_files" appended to it. For example, the pyramid for test.dzi is stored in test_files. Each level may be broken up into several tiles. The tiles are named as column_row.format, where row is the row number of the tile (starting from 0 at top) and column is the column number of the tile (starting from 0 at left). format is the appropriate extension for the image format used – either JPEG or PNG."

Please note, that "sparse images" and "collections" are not supported in the SZI format.

Thus, the Deep Zoom format is a pyramid-like directory structure, where each directory is one zoom level (and named by the number of the respective zoom level). The lowest number is 0 and contains the image in the size of 1 pixel. The highest number is the zoom level, which contains the image in the full resolution, in tiles.

3.2. Folder and file structure of the Deep Zoom pyramid (modified for SZI)

The Microsoft Deep Zoom pyramid was modified for the SZI format as described below. The whole pyramid, starting with a “root folder” will be introduced into an uncompressed ZIP file. Thus, when you open the SZI/ZIP file, you first and only see the root folder. The entire structure is depicted in the following figure.

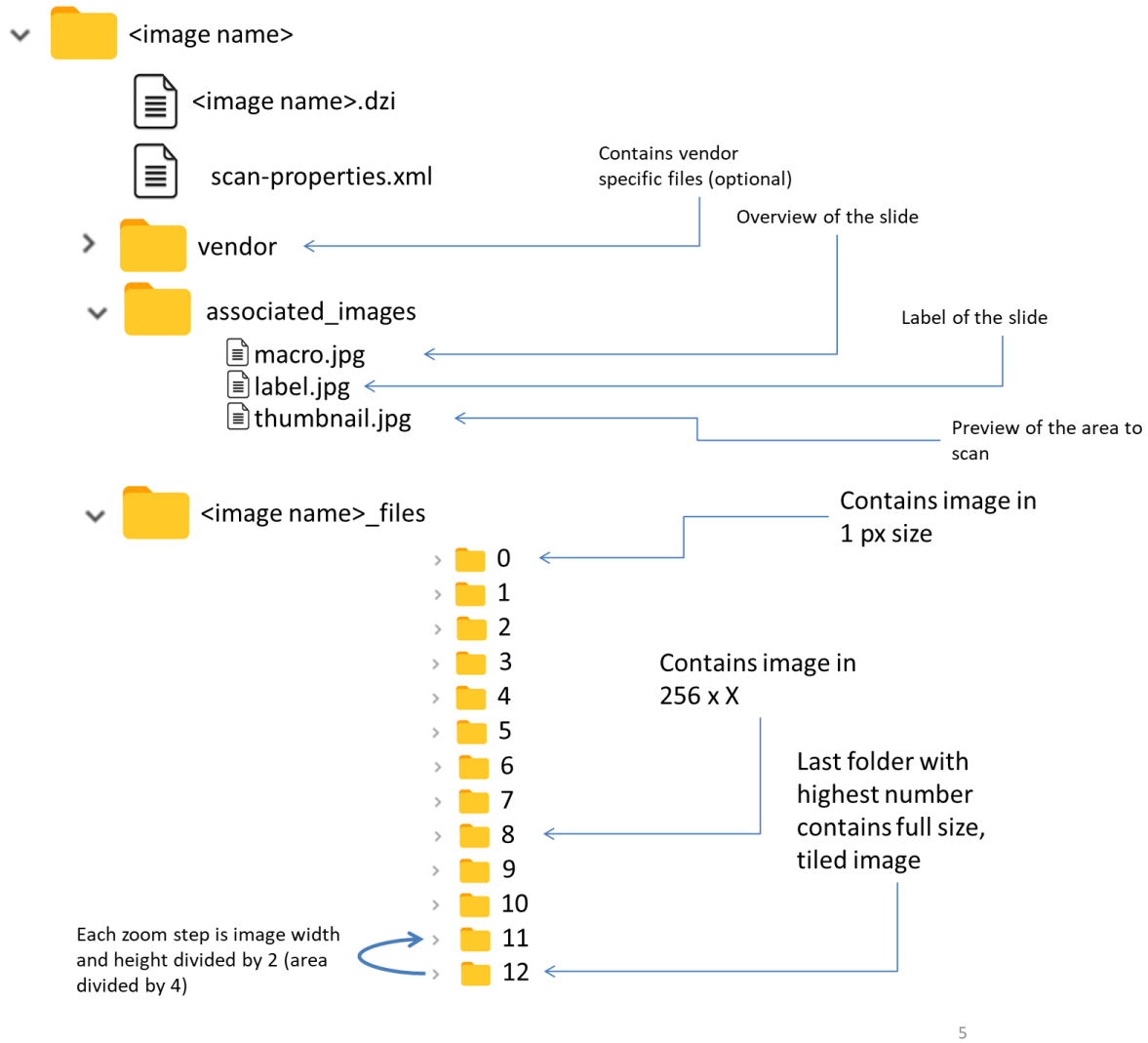

Figure. Folder structure of the Deep Zoom format. Note: the first folder in this figure is the root folder of the Deep Zoom pyramid. The entire folder structure is stored in the .szi file.

The root folder is labeled as the image name (e. g. CMU-1) and contains:

1. The .dzi file, which is is labeled as the image name plus the extension “.dzi”, e. g. CMU-1.dzi,
2. The scan-properties.xml file, which contains the scanning informations.
3. A folder, which is labeled as the image name plus “_files”, so in our example CMU-1_files. This contains the image pyramid.

4. A folder labeled “vendor” (optional), where all vendor specific files can be stored, which do not fit into any pattern of the SZI format.
5. A folder labeled “associated_images”

In the “_files” folder, we find further folders, which comprise the image pyramid. They are labeled “0” to the last and deepest zoom level, e. g. “12”. In the “_files” folder, there may also be the file scan-properties.xml, which contains scanning informations.

In the root folder, we also find an optional folder labeled “vendor”. Vendors can place any files and information, which do not fit any structure in the SZI into this folder to not lose any original information.

The root folder also contains a folder “associated_images” (optional for scanners with this functionality). In this folder, 3 images are stored:

- macro.jpg: This is an image which contains an overview of the whole glass, possibly extending beyond the edges of the glass
- label.jpg: This is an image of the label (e. g. paper label, or printed bar code label).
- thumbnail.jpg: This is an overview image of the scanned region.

By excluding the label and the macro from the overview image, the SZI can be anonymized by removing the label which may contain patient data.

The SZI format can also support the Zoomify and the Google Map format, but not in the current version. This functionality will be excluded from the current document.

To calculate the deep zoom directory, you have to perform the following:

Calculate the number of zoom levels: take the larger length of width and height and perform a log2, then ceil, then add +1 for DeepZoom --> these are the number of zoom levels from 0 to n together (e. g. result is 17, if zoom levels 0 to 16)

To present this functionality, a PHP script can be found in the appendix, which also calculates the tiles in each zoom level.

3.3.The DZI file

The Deep Zoom format contains a XML file with the extension ".dzi" in the main directory.

It contains informations about

- the XML format
- the tile image format (jpeg recommended), alternatively png
- the overlap (this stems from older times as an overlap of tiles may have given a better viewing experience. It is not necessary anymore, so always use "0", so tiles do not overlap)
- tile size (standard is 256, but can also vary. For performance, 256 can but should not be exceeded)
- height of image in pixels
- width of image in pixels.

An example of the DZI file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Image xmlns="http://schemas.microsoft.com/deepzoom/2008"
  Format="jpeg"
  Overlap="0"
  TileSize="256"
  >
  <Size
    Height="2967"
    Width="2220"
  />
</Image>
```

As XSD schema:

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
      <xs:element name="Image">
          <xs:complexType>
              <xs:sequence>
                  <xs:element name="Size">
                      <xs:complexType>
                          <xs:attribute name="Height" type="xs:int"></xs:attribute>
                          <xs:attribute name="Width" type="xs:int"></xs:attribute>
                      </xs:complexType>
                  </xs:element>
              </xs:sequence>
              <xs:attribute name="xmlns" type="xs:string"></xs:attribute>
```

```
            <xs:attribute name="Format" type="xs:string"></xs:attribute>
            <xs:attribute name="Overlap" type="xs:int"></xs:attribute>
            <xs:attribute name="TileSize" type="xs:int"></xs:attribute>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

3.4.The scan-properties.xml file

All scanners output scanning information / metadata like scan date, scan time, the used objective, µm/pixel, etc. Microscope scanning companies, which write the SZI from their own software, should create the file “scan-properties.xml”. The information of the layout of that file is provided further below and in the Appendix.

Of course your scanner will produce more meaningful data. We strongly encourage to add these data to the scan-properties.xml file. Just add your scanner name before the field name, separated by a dot “.”, e. g. “vendor.MicronsX” or “ScanCompany.FilterName”, etc.

Please input data into the following fields. The only mandatory fields are “ImageHeight”, “ImageWidth”, MicronsPerPixelX, MicronsPerPixelY, and “MicronsPerPixel” so that viewers can add measurement functionality.

Field in scan-properties.xml	**Value**	**Format**	**Example**
VendorName	Name of scanner vendor	string	Scan Co.
ScannerName	Name of scanner model	string	SuperScanner
ScannerSerialNo	Serial number of this scanner	string	2354H2kH
SoftwareName	Name of software used for scanning	string	ScanView
SoftwareVersion	Software version	string	2.0.5
UserName	Name of operator	string	Peter
TimeStart	Scan time start	yyyy-mm-ddThh:mm:ss (24 hour format)	2017-06-15T13:45:30
TimeEnd	Scan time end	yyyy-mm-ddThh:mm:ss (24 hour format)	2017-06-15T13:48:44
ElapsedTime	Scan time total	string with numbers in x, y, z: <x>h<y>m<z>s	0h3m22s
CaseNumber	Case number (dont enter patient name)	string	H2017-234
ScanJobName	Name of scan job	string	
ObjectiveMagnification	Objective used	int	E. g. 4 for 4x, 10 for 10x
CameraName	Name of the camera used	string	Sony
SensorPixelSize	Sensor Pixel size in µm	float	1.465
ScannedArea	Area of scan in mm²	float	566.7
ScanWidth	Width of scan in mm	float	12
ScanHeight	Height of scan in mm	float	11
ImageHeight	Height in pixel of resulting	int	23235

	scan		
ImageWidth	Width in pixel of resulting scan	int	39392
MicronsPerPixel	**Microns (µm) per pixel VERY IMPORTANT FIELD TO ADD A MEASUREMENT TOOL TO VIEWERS. If microns per X and Y do not match, take an average here**	float	0.4042
MicronsPerPixelX	**Microns (µm) per pixel on X**	float	0.4042
MicronsPerPixelY	**Microns (µm) per pixel on Y**	float	0.4042
Comments	Free text field	string	

An example of a scan-properties.xml file is in the Appendix.

3.5.Tiles in the Deep Zoom format

We recommend 256 x 256 tiles in jpeg for the SZI format. The tiles can be smaller, but for serving efficiency, they should not be much bigger than 256 x 256. Regarding the compression level of jpeg, Q=85 is a good compromise between size and quality.

In zoom levels 0 to 8, there is only one tile present. In zoom level 8, this tile's largest dimension (width or height) is between 129 and 256 pixels.

To match the entire image, border tiles (outermost right and bottom of the image) may have a different size from the standard tile to match the image width/height. E. g., if all tiles are 256 x 256, the right border tiles could be e. g. width x height: 122 x 256 or the lower right corner tile could be e. g. 122 x 56. as depicted in the graphic.

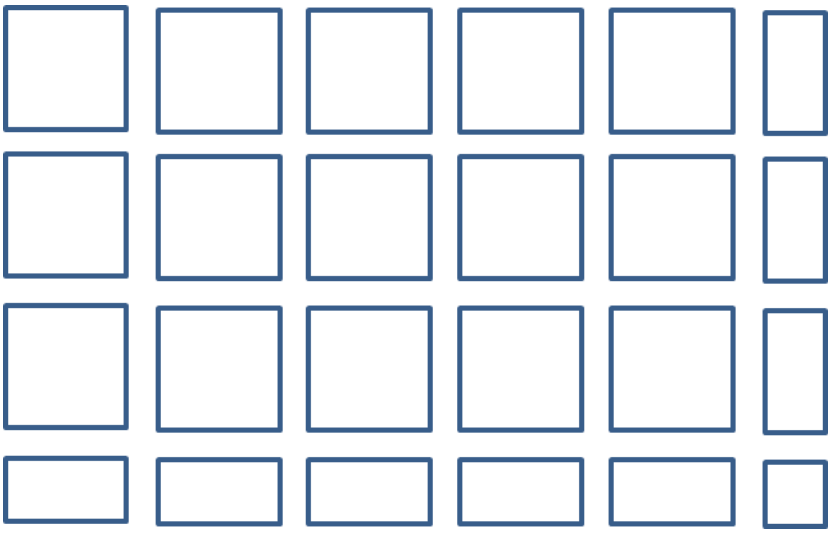

Figure. The tile layout could be like this with border tiles, which may have a different size from the standard tile size due to the image width/height.

4. The ZIP file

ZIP is a standardized file format which was invented by Phil Katz to store and compress files in order to save time and storage space.

For the SZI format, the entire Deep Zoom folders have to be written into an **UNCOMPRESSED** ZIP file. It is mandatory to switch off the compression as the serving of tiles out of the SZI would not work. The ZIP file should have the extension ".szi" so that viewers will recognize the ZIP files as an image file.

All folders + .dzi have to be stored in the ZIP file, so when opening the ZIP file, you should only see the root folder (named with the image name).

You can use any library for writing the ZIP file. There is no need to order the stored files in a certain way within the ZIP file. As the ZIP file can only store files up to 4 GB, images which are larger than 4 GB have to use ZIP64.

5. Creating an .szi file

5.1.From a scanning software

A scanning software can write the tiles directly into a ZIP file and rename this to .szi in the end. A .dzi file has to be included as well as a scan-properties.xml file (specifications will be given later about scan-properties.xml) and preferably the associated_images. Other information and files can be stored in the “vendor” folder, which is located in the root folder.

5.2.Working with VIPS

You only have to read this section, if you want to create SZI files from other scanner files through the OpenSlide library with (lib)VIPS. With VIPS you can convert e. g. Hamamatsu, Leica, Aperio, Philipps and other scanner files into SZI quickly..

VIPS is a very powerful open source library, which supports any kind of image manipulation of huge images. It was developed and is under continuous improvement by John Cupitt. VIPS has also integrated the OpenSlide library to work with microscope image formats from different vendors like Leica/Aperio, Hamamatsu, Mirax, etc.

With VIPS it is possible to convert any of these (OpenSlide supported) microscope images to the SZI format. Of course, VIPS can also convert all other supported image formats like JPEG, PNG, TIFF, JPEG2000, etc to SZI.

It is recommended to get familiar with VIPS and to convert some test files into the SZI format. To do this, VIPS has to be installed (it is mainly developed for Linux, but there are also Windows versions). If the command line tool / console is not desired, there is a Windows tiling program with a GUI by Smart In Media, which has integrated VIPS.

The function dzsave converts images into either tiles on the hard drive in the Deep Zoom format or into the SZI format. The command is:

```
vips.exe dzsave input.tiff output.szi --overlap=0 --
suffix=.jpg[Q=85] --tile-size=256 --properties --vips-progress
```

If you do not write any extension in the output-file, the result will be tiles on the hard drive. If you add “.zip” or “.szi”, it will be written into an SZI file automatically.

6. Appendix

6.1.Calculating the zoom levels in PHP

```
private function getOverviewCalculation($width, $height, $overlap,
$tilesize, $layout)
    {
        if (!isset($overlap)) $overlap = 0;
        if (!isset($tilesize)) $tilesize = 256;

        $wpixel = $width;
        $hpixel = $height;

        //Number of zoom levels: take the larger length of width and
height and make a log2, then ceil, then add 1 for dz, -7 for
zoomify--> these are the number of zoom levels from 0 to n together
(e. g. result is 17, if zoom levels 0 to 16)
        $maxlength = ($width > $height) ? $width : $height;
        switch ($layout) {
            case 1: //DeepZoom
                $total_zoomlevels = (ceil(log($maxlength, 2)) + 1);
                break;

            case 2: //Zoomify
                $total_zoomlevels = (ceil(log($maxlength, 2)) - 7);
                break;

            case 3: //Google Maps
                $total_zoomlevels = (ceil(log($maxlength, 2)) - 7);
                break;
        }
        $zoomlevel_info = array();

        //echo "<br />Number of Zoom levels:
".$total_zoomlevels."<br /><br />";
        for ($i = $total_zoomlevels - 1; $i >= 0; $i--) {
            $zoomlevel_info[$i] = array("number_of_tiles" =>
ceil($wpixel / $tilesize) * ceil($hpixel / $tilesize),
                "width" => $wpixel,
                "height" => $hpixel,
                "zoomlevel_no" => $i,
                "number_tiles_x" => ceil($wpixel / $tilesize),
                "number_tiles_y" => ceil($hpixel / $tilesize)
            );

            switch ($layout) {
                case 1:
                    $wpixel = ceil($wpixel / 2);
                    $hpixel = ceil($hpixel / 2);
                    break;
```

```
                case 2:
                    $wpixel = floor($wpixel / 2);
                    $hpixel = floor($hpixel / 2);
                    break;

                case 3:
                    $wpixel = ceil($wpixel / 2);
                    $hpixel = ceil($hpixel / 2);
                    break;
                default:
                    $this->error2tile('<br />Layout not correct');
            }
        }
        $cumulative_tiles = 0; //This is to later speed up the
calculation of the order of the tiles
        for ($i = 0; $i < $total_zoomlevels; $i++) {
            $zoomlevel_info[$i]['cumulative_number_tiles_before'] =
$cumulative_tiles;
            $cumulative_tiles +=
$zoomlevel_info[$i]['number_of_tiles'];
        }
        return $zoomlevel_info;
 }
```

6.2.The scan-properties.xml file

The structure of the scan-properties.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="image">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="properties">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="property"
maxOccurs="unbounded">
								<xs:complexType>
									<xs:sequence>
										<xs:element
name="name" type="xs:string"></xs:element>
										<xs:element
name="value" type="xs:string"></xs:element>
									</xs:sequence>
								</xs:complexType>
							</xs:element>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
			</xs:sequence>
			<xs:attribute name="xmlns" type="xs:string"></xs:attribute>
			<xs:attribute name="date" type="xs:date"></xs:attribute>
			<xs:attribute name="version" type="xs:string"></xs:attribute>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

An example file:

```
<?xml version="1.0"?>
<image xmlns="http://www.pathozoom.com/szi" date="2017-10-16"
version="1.0">
  <properties>
     <property>
          <name>VendorName</name>
          <value>TestCompany</value>
     </property>
     <property>
          <name>ScannerName</name>
          <value>Super Scan 2</value>
     </property>
```

```
<property>
      <name>ScannerSerialNo</name>
      <value>230.35-i38u3</value>
</property>
<property>
      <name>SoftwareName</name>
      <value>Scan it</value>
</property>
<property>
      <name>SoftwareVersion</name>
      <value>5.3</value>
</property>
<property>
      <name>UserName</name>
      <value>thomas</value>
</property>
<property>
      <name>TimeStart</name>
      <value>2017-10-16T15:33:31</value>
</property>
<property>
      <name>TimeEnd</name>
      <value>2017-10-16T15:50:53</value>
</property>
<property>
      <name>ElapsedTime</name>
      <value>0h17m22s</value>
</property>
<property>
      <name>CaseNumber</name>
      <value>H-2017-234</value>
</property>
<property>
      <name>ScanJobName</name>
      <value>Test-Scan</value>
</property>
<property>
      <name>ObjectiveMagnification</name>
      <value>10</value>
</property>
<property>
      <name>CameraName</name>
      <value>Basler</value>
</property>
<property>
      <name>SensorPixelSize</name>
      <value>1.453</value>
</property>
<property>
      <name>ScannedArea</name>
      <value>556.7</value>
</property>
<property>
      <name>ScanWidth</name>
      <value>12</value>
</property>
```

```
      <property>
            <name>ScanHeight</name>
            <value>11</value>
      </property>
      <property>
            <name>ImageHeight</name>
            <value>2967</value>
      </property>
      <property>
            <name>ImageWidth</name>
            <value>2220</value>
      </property>
      <property>
            <name>MicronsPerPixel</name>
            <value>0.402</value>
      </property>
      <property>
            <name>MicronsPerPixelX</name>
            <value>0.402</value>
      </property>
      <property>
            <name>MicronsPerPixelY</name>
            <value>0.402</value>
      </property>
      <property>
            <name>Comments</name>
            <value>This is a test comment!</value>
      </property>

   </properties>
</image>
```