LightField®

Scientific Imaging and Spectroscopy Software

SPE 3.0 File Format Specification

Princeton Instruments

SPE 3.0 File Format

Specification

Princeton Instruments

4411-0140
Issue 3
January 2, 2018

Revision History

Issue	Date	List of Changes
3	January 2 2018	Issue 3 of this document incorporates the following changes: • Added the Revision History table; • Updated the copyright year.
2	January 8, 2016	Issue 2 of this document incorporates the following changes: • Updated the copyright year.
1.A	January 21, 2014	Issue 1.A of this document incorporates the following changes: • Updated the copyright year.
1	May 2, 2012	This is the initial release of this document.

Princeton Instruments, a division of Roper Scientific, Inc.
3660 Quakerbridge Rd.
Trenton, NJ 08619
TEL: 800-874-9789 / 609-587-9797
FAX: 609-587-1970
www.princetoninstruments.com

Table of Contents

Figures

Tables

Chapter 1

Introduction to SPE 3.0

Introduction

The primary purpose of the SPE file format is to store scientific imaging data. SPE files can also optionally contain experiment information describing how the data was obtained as well as a history of data processing that was applied to the data. SPE 2.x is the native file format for WinView/WinSpec, while SPE 3.0 is that for LightField.

Manual Organization

The primary focus of this manual is to describe terms and concepts related to the SPE 3.0 format. Examples and suggestions are provided to enhance understanding.

Chapter 1: Introductory information about the nature of SPE, the differences between SPE 2.x and SPE 3.0 file formats, the data types used in SPE 3.0, and the major sections of an SPE 3.0 file.

Chapter 2: Discussion of data extraction, including three examples.

Chapter 3: Discussion of what metadata is and how to access in an SP 3.0 file.

Chapter 4: Discussion of calibrations.

Chapter 5: Discussion of experiment and data processing information contained in an SP 3.0 file.

Chapter 6: Discussion of miscellaneous information that may be contained in an SP 3.0 file.

Appendix A: Provides SPE 2.x header structure details.

What's New in SPE 3.0

Whereas SPE 2.x is structured as a fixed-size binary header followed by binary image data, SPE 3.0 allows for more features and extensibility than previous versions by including an XML footer following the binary image data.

The SPE 2.x header defines the shape and type of the image data and optionally describes some experiment parameters used to acquire the data. With SPE 3.0, the fixed binary header has been largely replaced by an XML footer which provides the following benefits:

- Information can be added unbounded since the footer size is not fixed.
- Information that can be naturally grouped or structured can be better represented.
- Image data can take complex shapes allowing supplemental data to be associated with the image data (metadata).
- Custom information can be added without impacting other software.

Basic SPE Structure

As previously stated, SPE 2.x is structured as a fixed-size binary header immediately followed by binary image data and SPE 3.0 extends this structure to include an XML footer following the binary image data (see Figure 1).

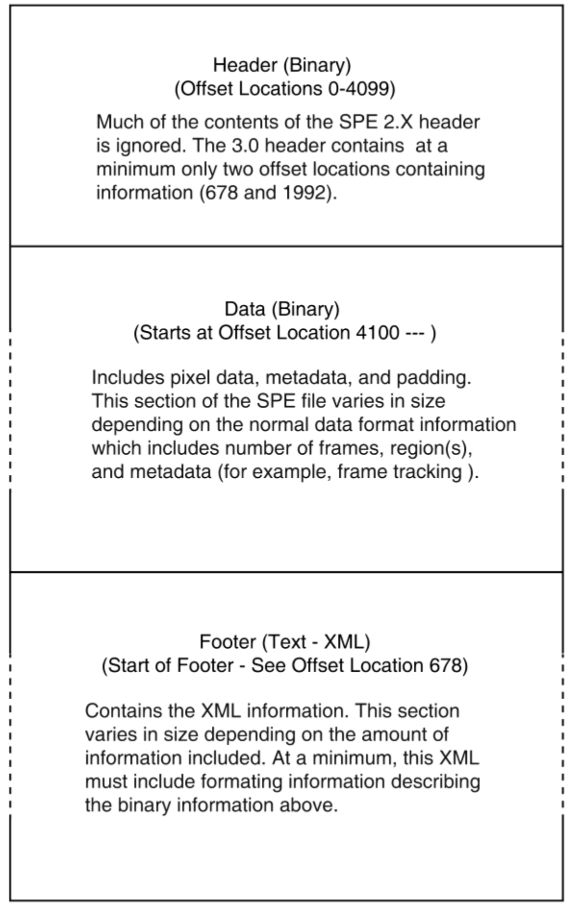

Figure 1. SPE 3.0 File Structure

Defined Data Types

Note that all binary data is little-endian.

Binary Type	Description
8s	8-bit signed integer
8u	8-bit unsigned integer
16s	16-bit signed integer
16u	16-bit unsigned integer
32s	32-bit signed integer
32u	32-bit unsigned integer
64s	64-bit signed integer
64u	64-bit unsigned integer
32f	32-bit floating point
64f	64-bit floating point

Table 1. Binary Data Types (64-bit)

Binary Header Section

The binary header is a fixed size of 4,100 bytes. Each field is a particular binary type and located at an offset from the beginning of the file in bytes.

Originally, the fixed-size header from SPE 2.x described both the type and shape of the image data along with optionally including experiment or processing details. The following restrictions apply to such a header:

- Each frame must be a rectangle of pixels. Multiple regions are supported, but they must still form a rectangle of data within the frame (usually by including “filler” pixels to form a rectangular shape).
- Optional details can only be defined by Princeton Instruments. Through the years, many of these have been phased out but still remain for legacy reasons.
- The fixed-size of the header puts an upper bound on the amount of additional content the header could support, which makes associating information that varies per-frame impossible to store.

SPE 3.0 removes these limitations by deprecating the majority of the header and replacing it with an XML footer. The following fields are required (all others are optional and can be initialized to 0):

Binary Type	Name	Offset	Description
32f	file_header_ver	1992	SPE version
64u	xml_footer_offset	678	offset to the XML footer in bytes

Table 2. Header Fields required for SPE 3.0

Other fields may be optionally set depending on the level of backwards-compatibility with SPE 2.x that one wishes. For instance, if the type of and shape of the image data meets the requirements of SPE 2.x, the following fields can additionally be set for backwards compatibility with software reading SPE 2.x image data (such as WinView or WinSpec; in fact this is precisely the level of backwards-compatibility obtained by SPE files created by LightField):

Binary Type	Name	Offset	Description
16s	datatype	108	binary type of pixel
16u	xdim	42	width of a frame in pixels
16u	ydim	656	height of a frame in pixels
32s	NumFrames	1446	number of frames
16u	xDimDet	6	set to xdim (required for legacy reasons)
16u	yDimDet	18	set to ydim (required for legacy reasons)
16s	noscan	34	set to -1 (required for legacy reasons)
32s	lnoscan	664	set to -1 (required for legacy reasons)
16s	scramble	658	set to 1 (required for legacy reasons)
32s	WinView_id	2996	set to 19,088,743 (or 1234567 hex) (required for legacy reasons)
16s	lastvalue	4098	set to 21,845 (or 5555 hex) (required for legacy reasons)

Table 3. Additional Header Fields required for SPE 2.x Compatibility
(Image data must meet type and shape requirement of SPE 2.x.)

The *datatype* field can be one of the values in Table 4.

Note: Values denoted by (SPE 2.x only) are not supported for SPE 3.0.

Pixel Type	datatype Value
8u	6 (SPE 2.x only)
16u	3
16s	2 (SPE 2.x only)
32u	8
32s	1 (SPE 2.x only)
32f	0
64f	5 (SPE 2.x only)

Table 4. datatype Values supported for SPE 2.x

Binary Data Section

Image data is stored in binary as groupings of pixels where each pixel holds a monochromatic intensity. Working from microscopic to macroscopic, pixels are grouped into rows; rows into regions and finally regions into frames. More specifically, if the file is read in consecutive bytes per pixel, the first pixel represents the intensity at the top-left corner of the first region of interest (ROI) for the first frame. The next pixel would be in the same row but one column to the right, and so on until the *width*-number of pixels has been read to complete the row. The next pixel read starts the next row of the first region for the first frame. This then repeats until *height*-number of rows have been read to complete the first frame of the first region. Ordering a single image in this way is commonly called raster order.

Width-Number of Pixels = 6

Height-Number of Rows = 7

X0Y0	X1Y0	X2Y0	X3Y0	X4Y0	X5Y0
X0Y1	X1Y1	X2Y1	X3Y1	X4Y1	X5Y1
X0Y2	X1Y2	X2Y2	X3Y2	X4Y2	X5Y2
X0Y3	X1Y3	X2Y3	X3Y3	X4Y3	X5Y3
X0Y4	X1Y4	X2Y4	X3Y4	X4Y4	X5Y4
X0Y5	X1Y5	X2Y5	X3Y5	X4Y5	X5Y5
X0Y6	X1Y6	X2Y6	X3Y6	X4Y6	X5Y6

First row read: X0Y0, X1Y0, ... X5Y0

Next row read: X0Y1, X1Y1, ... X5Y1

Figure 2. ROI Data Layout in Raster Order

SPE 3.0 supports any number of regions of interest for any number of frames. This lifts the previous requirement of SPE 2.x where frames must be rectangular and multiple regions typically required “filler” pixels to form a rectangle. However, the following requirements do apply:

- Pixels within a region of interest have the same binary type.
- All regions of interest have the same pixel binary type.
- Regions of interest are rectangular (i.e., have a width and a height).
- All frames have the same regions of interest in the same order.

SPE 3.0 allows additional data to be stored in binary besides image data. Called metadata, it is typically data that varies per region or per frame. Metadata may be appended to any region per frame or to each frame. Additionally, binary padding may be applied to any region per frame or to each frame, but must be last (following the appropriate image data and/or metadata).

Note: LightField only supports metadata per frame.

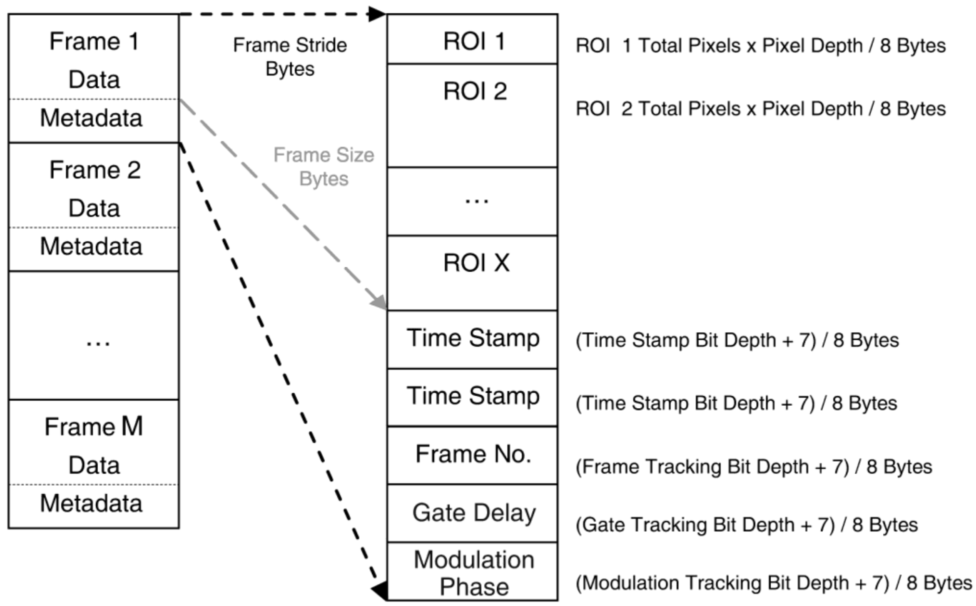

Figure 3. LightField Data Format Diagram

Figure 3 shows a graphical representation of the LightField binary data format. Among the terms used in the drawing are frame stride, frame size, ROI, five types of metadata. These terms are also used below in describing the binary data structure that starts at offset 4100: all partitions are specified in bytes.

- A frame stride includes the frame pixel data and any frame metadata and/or padding.
- Frame pixel data contains data for X regions of interest (ROIs); whose regions are in the order in which each region was defined.
- Frame size is the sum of the (width x height x pixel size) of all ROIs in the frame. Pixel size depends on the pixel data type (pixelFormat) and will be either 2 or 4. If for example, if there were two ROIs one of 175 x 1 and the other 125 x 1 and the pixel format was MonochromeUnsigned16 (2 bytes per pixel), the frame size would be (175 x 1 x 2) + (125 x 1 x 2) or 600 bytes.

  The following pixel data types are supported:

  - MonochromeUnsigned16 (2 bytes)
  - Monochrome Unsigned32 (4 bytes)
  - MonochromeFloating32 (4 bytes)
- Frame metadata contains any time stamps, frame tracking, gate tracking, gate tracking, and/or modulation tracking information associated with the data.
  - If there are no frame metadata, the values of frame size and frame stride will be identical.
  - If there is frame metadata, the frame stride will exceed the frame size by 8 bytes per metadata type. For example, assuming a frame size of 300 bytes and the inclusion of exposure start and exposure end, the frame stride would be 300 + 8 + 8 or 316 bytes.

Data has the following layout:

- One frame of image data containing each region of interest (in the order defined)
- Followed by defined metadata for that frame (any combination of timestamps, frame tracking, gate tracking, gate tracking, and modulation tracking)
- Repeated for each frame.

The details of reading data and/or metadata can be found in the appropriate chapters.

XML Footer Section

This section follows the binary data section in the file. The footer content is a valid XML 1.0 document stored in a valid Unicode encoding (typically UTF-8). All XML elements that are part of the SPE format belong to the **http://www.princetoninstruments.com/spe/2009** XML namespace. Furthermore, all elements in this namespace (and attributes of these elements) will be in an invariant locale (very similar to the en-us format; American English). Unless otherwise noted, these elements and attributes can be in any order.

The XML footer of any SPE 3.0 file can be extracted by LightField by opening the file, showing file information, and saving the information to an XML file.

The root element is **SpeFormat** and contains a **version** attribute that states the SPE version (which must match the value of *file_header_ver* in the binary header). The following child elements are defined in SPE 3.0:

- **DataFormat** **(required):** describes the layout and type of image data in the binary data section.
- **MetaFormat** **(possibly required):** describes the layout and type of metadata in the binary data section (if any).
- **Calibrations** **(possibly required):** contains any applicable calibration associated with the image data.
- **DataHistories** **(optional):** provides information on the experiment used to acquire and/or process the image data.
- **GeneralInformation** **(optional):** provides miscellaneous information about the file.

A collapsed view of these XML elements is shown in Figure 4. An example of the minimum elements required for an SPE 3.0 file is shown in Figure 5.

```
  <?xml version="1.0" encoding="utf-8" ?> *
- <SpeFormat version="3.0" xmlns="http://www.princetoninstruments.com/spe/2009">
  + <DataFormat>
  + <MetaFormat>  *
  + <Calibrations> *
  + <DataHistories> **
  + <GeneralInformation> **
 </ SpeFormat >
```

* The MetaFormat and Calibrations elements may be required.

** The <?xml version…>, DataHistories, and GeneralInformation elements are optional.

Figure 4. Collapsed View of XML Elements in Footer

```xml
<SpeFormat version="3.0" xmlns="http://www.princetoninstruments.com/spe/2009">
    <DataFormat>
      <DataBlock type="Frame"
                 count="5"
                 pixelFormat="MonochromeUnsigned16"
                 size="176400"
                 stride="176400">
          <DataBlock type="Region"
                     count="1"
                     width="210"
                     height="320"
                     size="134400"
                     stride="134400" />
          <DataBlock type="Region"
                     count="1"
                     width="210"
                     height="100"
                     size="42000"
                     stride="42000" />
      </DataBlock>
    </DataFormat>
</SpeFormat
```

Figure 5. Example 1: Minimum XML required for SPE 3.0 File

In Figure 5 `<SpeFormat version="3.0" xmlns="http://www.princetoninstruments.com/ spe/2009">` must be included since it indicates the SPE version and defines the SPE XML namespace . **DataFormat** and its children describe the data and the size of the dataset.

- The Frame **DataBlock** reports the number of frames (count), the **pixelFormat** (which plays a part in the calculation of size and stride), and the size and stride.
- The Region **DataBlock** describes the ROIs in the frame in a bare bones fashion. How many different ROIs of that size there are (count) (will always be 1, for now), the width and height of those ROIs, and the calculated size and stride. If **pixelFormat="MonochromeUnsigned16"**, then for a region multiply width x height x 2 to get **size**. If **pixelFormat** were **monochromeUnsigned32** or **monochromeFloating32**, then **size** would be width x height x 4.
- Notice how the ROI sizes add up to the Frame size. The ROI strides add up to the Frame stride. In the example above, the sizes and strides match. This changes when metadata is associated with a Frame.

Notes:

1. The **calibrations** attribute (not shown in Figure 5) is not required for a Region **`DataBlock`** in the bare minimum XML. However, if it is included, the **`Calibrations`** element must be included in the XML.
2. The **metaFormat** attribute (also not shown) is not required for a Frame **`DataBlock`** in the bare minimum XML. However, if it is included, the **`MetaFormat`** element must be included in the XML.

As a general rule, any custom XML can be added provided:

- All custom elements are not in the SPE XML namespace.
- Any custom attributes applied to a SPE element are in a namespace (and not the SPE XML namespace).

This page intentionally left blank.

Chapter 2

Accessing Data

Introduction

A single **DataFormat** element describes type and layout of image data. It does this using a hierarchy of **DataBlock** elements. The first **DataBlock** must be a child of **DataFormat** and describe frames with the attributes in Table 1 below.

Attribute	Value
type	Frame
count	number of frames
pixelFormat	pixel type of all pixels in all frames*
size	total number of bytes required to store all pixels in all regions in one frame
stride	total number of bytes to skip to get to the beginning of the next frame from the start of a frame

Table 5. Frame DataBlock Attributes

* SPE 3.0 supports the following pixel binary types:

Pixel Type	pixelFormat	datatype
16u	MonochromeUnsigned16	3
32u	MonochromeUnsigned32	8
32f	MonochromeFloating32	0

Table 6. Pixel Binary Types

Each region of interest within a frame is described with a corresponding child **DataBlock** of the Frame **DataBlock** with attributes as follows:

Attribute	Value
type	Region
count	1
width	width of the region in pixels
height	height of the region in pixels
size	total number of bytes required to store all pixels in this region
stride	total number of bytes to skip to get to the beginning of the next region from the start of this region

Table 7. Region of Interest DataBlock Attributes

The ROI sizes add up to the Frame size. The ROI strides add up to the Frame stride. When there is no metadata associated with a Region **DataBlock**, the sizes and strides match. The size and stride will no longer match when one or more pieces of metadata are associated. Currently, the metadata that could be associated include: Exposure Started, Exposure Ended, Frame Tracking, Gate Tracking, and Modulation

Tracking. If metadata are associated, an additional attribute **MetaFormat** will be added to the Frame **DataBlock**. Each piece of metadata associated with a region adds 8 bytes to the Frame stride.

Because metadata is per frame information that is stored in line with the pixel data (pixel data and metadata cover all of the binary data), it is very important to use **stride** when extracting data from a file. Of the three examples provided in this chapter, only Example 3 contains metadata.

Data Extraction

Simple Data - Single Region with Multiple Frames

The example in Figure 6 is used to demonstrate a simple data extraction. Note that properties irrelevant to data extraction have been omitted.

```
<DataFormat>
    <DataBlock type="Frame"
               count="5"
               pixelFormat="MonochromeUnsigned16"
               size="134400"
               stride="134400">
    <DataBlock type="Region"
               calibrations="1,2"
               count="1"
               width="210"
               height="320"
               size="134400"
               stride="134400" />
    </DataBlock>
</DataFormat>
```

Figure 6. Example 2: Simple Data – Single Region, Multiple Frames, No Metadata

1. The example shows that there are 5 frames of 16-bit data with a region (or image size) of 210 x 320 pixels. In both of these **DataBlock** elements, the **size** attribute indicates the number of bytes of 1 item (region or frame, indicated by the **type** attribute), while **stride** indicates the relative offset to the next item. With simple data as above, **size** will equal **stride**.

2. With this information, the first frame is located at offset 4100; the second frame at offset 4100+134400=138500; the third frame at offset 138500+134400=272900; and so on.

Extracting Complex Data – Multiple Regions

The example in Figure 7 shows an XML fragment from an SPE file containing 5 frames and 3 ROIs (properties not relevant to data extraction are omitted).

```xml
<DataFormat>
    <DataBlock type="Frame"
               count="5"
               pixelFormat="MonochromeUnsigned16"
               size="294202"
                stride="294202">
        <DataBlock type="Region"
                   count="1"
                   width="210"
                   height="320"
                    size="134400"
                   stride="134400" />
        <DataBlock type="Region"
                   count="1"
                   width="236"
                   height="338"
                   size="159536"
                   stride="159536" />
        <DataBlock type="Region"
                   count="1"
                   width="133"
                   height="1"
                   size="266"
                   stride="266" />
    </DataBlock>
</DataFormat>
```

Figure 7. Example 3: Complex Data – Multiple Regions, Multiple Frames, No Metadata

1. The file indicates 5 frames of 16-bit data with 3 regions of 210 x 320, 236 x 338, and 133 x 1 pixels (in that order). Each frame contains all three regions; note the frame size is the sum of each region size. In this case, the region stride moves from one region to the next region, while the frame stride moves from one frame to the next.
2. With this information, the first frame of region 1 is located at offset 4100; region 2 at offset 4100+134400=138500; region 3 at offset 138500+159536=298032. To find frame 2 region 2 one can navigate as follows: find the beginning of region 2 and then move a frame stride: 4100 (start of data) + 134400 (region 1 stride to region 2) +294202 (frame stride) = 432702.

Extracting Complex Data – Multiple Regions and Metadata

The example in Figure 8 shows an XML fragment from an SPE file containing 5 frames and 3 ROIs and 2 pieces of metadata (**ExposureStarted** and **Exposure Ended**). Properties not relevant to data extraction are omitted.


```xml
<DataFormat>
    <DataBlock type="Frame"
               count="5"
               pixelFormat="MonochromeUnsigned16"
               size="294202"
                stride="294218">
               metaFormat ="1"
        <DataBlock type="Region"
                   count="1"
                   width="210"
                   height="320"
                    size="134400"
                   stride="134400" />
        <DataBlock type="Region"
                   count="1"
                   width="236"
                   height="338"
                   size="159536"
                   stride="159536" />
        <DataBlock type="Region"
                   count="1"
                   width="133"
                   height="1"
                   size="266"
                   stride="266" />
    </DataBlock>
</DataFormat>
<MetaFormat>
    <MetaBlock id="1"
        <TimeStamp event="ExposureStarted"
                   type="Int64"
```

```
                         bitDepth="64"
                         resolution="2208037"
                         absoluteTime="2012-04-02T14:07:54.8046287-
                                        04:00"/>
            <TimeStamp event="ExposureEnded"
                         type="Int64"
                         bitDepth="64"
                         resolution="2208037"
                         absoluteTime="2012-04-02T14:07:54.8046287-
                                        04:00" />
    </MetaFormat>
```

Figure 8. Example 4: Complex Data – Multiple Regions, Multiple Frames, Metadata

1. The file indicates 5 frames of 16-bit data with 3 regions of 210 x 320, 236 x 338, and 133 x 1 pixels (in that order). Each frame contains all three regions; note the frame size is the sum of each region size and 8 bytes per piece of metadata. Region stride moves from one region to the next region, while the frame stride moves from one frame to the next.
2. With this information, the first frame of region 1 is located at offset 4100; region 2 at offset 4100+134400=138500; region 3 at offset 138500+159536=298032. To find frame 2 region 2 one can navigate as follows: find the beginning of region 2 and then move a frame stride: 4100 (start of data) + 134400 (region 1 stride to region 2) +294218 (frame stride) = 432718.

This page intentionally left blank.

Chapter 3
Accessing Metadata

Introduction

Metadata is supplemental binary data associated with image data. It can be associated per frame (metadata follows each frame) or per region per frame (metadata follows a region for every frame). Metadata is best used with data that varies per frame (or per region per frame). Supplemental data associated with image data that does not vary is better suited as calibration (described in the next chapter).

Notes:
1. LightField only supports per frame metadata.
2. The XML footer of any SPE 3.0 file can be extracted by LightField by opening the file, showing file information and saving the information to an XML file.

A single **MetaFormat** element describes type and layout of all metadata and is only required if metadata is present in the binary image data section. There will be one **MetaBlock** child (of the MetaFormat element) for each **DataBlock** that has associated metadata. The type of DataBlock determines if the metadata is per frame (**MetaBlock** type is **Frame**) or per region per frame (**MetaBlock** type is **Region**). A **MetaBlock** is associated with a **DataBlock** by assigning an **id** attribute on the **MetaBlock** and then referencing that **id** in the **metaFormat** attribute of the **DataBlock**.

A **MetaBlock** contains a child element for each piece of metadata. Which element depends on the type of metadata being represented. Below are the metadata defined in SPE 3.0. Custom metadata may be added using custom elements. However, any custom element must include a stride attribute whose value indicates the number of bytes required to skip this piece of metadata and a count attribute if multiple metadata values are contiguous.

All SPE 3.0 metadata has a type defining the binary type. The following types are defined:

type Value	Binary Type
Int64	64s
Double	64f

Table 8. SPE 3.0 Metadata Type Value

Metadata Elements

TimeStamp

The **TimeStamp** element describes a moment in time in ticks and has the following attributes:

Attribute	Definition
event	which event occurred at this time stamp
type	the binary type of the time stamp
bitDepth	the number of bits used in the binary type
resolution	the resolution of the time stamp in ticks per second
absoluteTime	the absolute time when the ticks were zero

Table 9. TimeStamp Attributes

The value of event can be one of the following:

event Value	Definition
ExposureStarted	exposure time has begun
ExposureEnded	exposure time has ended

Table 10. TimeStamp Component Value

FrameTrackingNumber

The **FrameTrackingNumber** element numbers the frame from the start of a continuous acquisition and has the following attributes:

Attribute	Definition
type	the binary type of the frame tracking number
bitDepth	the number of bits used in the binary type

Table 11. FrameTrackingNumber Attributes

GateTracking

The **`GateTracking`** element describes the value of a gating pulse component used to acquire the associated data and has the following attributes:

Attribute	Definition
component	which gate pulse component
type	the binary type of the component
bitDepth	the number of bits used in the binary type
monotonic (optional)	set to True if the value of the gate pulse component is always increasing, decreasing or equal; set to False if not; if this attribute is not present the component could be monotonic or not.

Table 12. GateTracking Attributes

The value of component can be one of the following:

component Value	Definition
Delay	the delay of the gate pulse in nanoseconds
Width	the width of the gate pulse in nanoseconds

Table 13. Gate Pulse Component Value

ModulationTracking

The **`ModulationTracking`** element describes the value of an RF modulation component used to acquire the associated data and has the following attributes:

Attribute	Definition
component	which RF modulation component
type	the binary type of the component
bitDepth	the number of bits used in the binary type
monotonic (optional)	set to True if the value of the gate pulse component is always increasing, decreasing or equal; set to False if not; if this attribute is not present the component could be monotonic or not.

Table 14. ModulationTracking Attributes

The value of component can be one of the following:

component Value	Definition
Phase	the phase of the RF modulation with respect to the user RF output in degrees

Table 15. RF Modulation Component Value

Example of Metadata in the XML Footer

```
<DataFormat>
    <DataBlock type="Frame"
                count="5"
                pixelFormat="MonochromeUnsigned16"
                size="147350"
                 stride="147374">
                metaFormat ="1"
        <DataBlock type="Region"
                    calibrations="1,2"
                    count="1"
                    width="180"
                    height="314"
                     size="113040"
                    stride="113040" />
        <DataBlock type="Region"
                    calibrations="1,3"
                    count="1"
                    width="235"
                    height="73"
                    size="34310"
                    stride="34310" />
    </DataBlock>
</DataFormat>
<MetaFormat>
    <MetaBlock id="1"
        <TimeStamp event="ExposureStarted"
                    type="Int64"
                    bitDepth="64"
                    resolution="2208037"
                    absoluteTime="2012-04-02T14:07:54.8046287-
                                  04:00"/>
         <TimeStamp event="ExposureEnded"
                    type="Int64"
                    bitDepth="64"
                    resolution="2208037"
                    absoluteTime="2012-04-02T14:07:54.8046287-
                                  04:00" />
        <FrameTrackingNumber type="Int64"
                                bitDepth="64" />
    </MetaFormat>
```

Figure 9. Example 5: Complex Data – Multiple Regions, Multiple Frames, Metadata

Chapter 4
Applying Calibrations

Calibration data is supplemental data associated with a region or frame that does not vary. "Calibrations" as used here is **NOT** referring specifically and ONLY to the kind of calibration used with spectrometers. The calibration values noted under the **`DataBlock`** (Frame) and **`DataBlock`** (Region) refer to the **id** numbers under the **`Calibrations`** element. **`WavelengthMapping`** applies to the Frame **`DataBlock`**. **`SensorInformation`** and **`SensorMapping`** apply to the Region **`DataBlock`**.

Here are the details for the different calibrations supported in SPE 3.0.

1.) Any calibration is optional – there can be zero or more calibrations that apply to different pieces of data.

2.) If a **calibrations** attribute is shown for a **`DataBlock`** element, the **`Calibrations`** element must be included in the XML.

3.) In the example shown in Figure 10, attributes previously covered have been removed.
   a. Any calibration is forward-referenced using the **calibrations** attribute on the **`DataBlock`** element to which it refers.
   b. The attribute contains one or more **id** numbers (comma delimited).
   c. Each **id** number refers uniquely to a child element of the **`Calibrations`** element.
   d. In this example, two calibrations apply to the region, while one applies to the frame.
   e. The child element defines the type of calibration.
   f. The following applies to the **`WavelengthMapping`** calibration:
      - There is an optional **date** attribute that denotes the date the wavelength calibration was performed.
      - There is an optional **orientation** attribute (similar to the one in **`SensorInformation`**), that defines the frame of reference for calibration.
      - This element must have one of the following elements as its child:
        1. The **`Wavelength`** element contains comma-delimited floating point numbers each mapping a column on the sensor to a wavelength (in nanometers).
        2. The **`WavelengthError`** element (not shown in Figure 10) contains whitespace-delimited wavelength/error pairs each mapping a column on the sensor to a wavelength (in nanometers) and an error (in delta nanometers). The pairs themselves are delimited by commas.

   g. The following applies to the **`SensorInformation`** calibration:
      - The **orientation** attribute defines the logical orientation of the sensor. It can any one of the following:
        1. **Normal** - the default orientation where the origin is the top-left corner
        2. One or more of the following (comma delimited):
           a. **FlippedHorizontally** – the sensor is reflected from left to right of **Normal**.
           b. **FlippedVertically** – the sensor is reflected from top to bottom of **Normal**.
           c. **RotatedClockwise** – the sensor is rotated clockwise in relation to **Normal**.
        3. This leads to one of eight possible geometries.
        4. Rotation is always applied after any flips. Another point of view is that rotation rotates the axes of symmetry for reflection as well.

- The **height** and **width** attributes define the logical dimensions of the sensor (in pixels). All regions are a subspace of this area. The **height** and **width** attributes are always defined relative to **Normal** orientation.

h. The following applies to the **SensorMapping** calibration:

1. The **x** and **y** attributes describe the top-left corner on the sensor (zero-based).
2. The **height** and **width** attributes describe the size of the region on the sensor in pixels.
3. The **xBinning** and **yBinning** attributes describe the combination of columns and rows on the sensor in relation to image data. This point of view is after the orientation is applied to the sensor.

```
<DataFormat>
   <DataBlock type="Frame"
              count="1"
              calibrations ="1">
   <DataBlock type="Region"
              calibrations="2,3" />
   </DataBlock>
</DataFormat>
                              -
                              -
                              -
<Calibrations>
   <WavelengthMapping id="1"
                      date="2012-01-18T10:59:53.025023-
                            05:00"
                      orientation ="Normal">
      <Wavelength xml:space="preserve">864.988931802598,
                      865.255537805099,865.522140883637,865
                      ...
      </Wavelength>
   </WavelengthMapping>
   <SensorInformation id="2"
                      orientation="Normal"
                      height="512"
                      width="512" />
   <SensorMapping id="3"
                  x="0"
                  y="254"
                  height="3"
                  width="512"
                  xBinning="1"
                  yBinning="3" />
</Calibrations>
```

Figure 10. Example 6: Calibrations Element

Chapter 5

Noting Experiment/ Processing Information

The next piece of the SPE 3.0 XML footer is the **DataHistories** element which chronicles supplemental details about how the data was taken and modified over time. The **Origin** information is captured when a SPE is created in LightField. As post-processes are performed in LightField, the user, the date the post-process was performed, and what was performed are captured.

1.) This optional section describes the birth of the data (origin) as well as any post-processing that has occurred.
2.) Often the origin information is orthogonal to any modifications that have occurred via post-processing.
3.) In the example shown in Figure 11, some elements and attributes have been removed:

```xml
<DataHistories>
   <DataHistory >
      <Origin creator="jjones"
              created="2012-02-14T15:46:39.1183649-05:00"
              software="LightField"
              softwareVersion="4.2.0.0 (Beta)"
              softwareCompany="Princeton Instruments">
                        (additional elements removed)
      </Origin>
      < DataModified user="jjones"
                     date="2012-04-30T13:54:26.9544718-04:00"
                     software="LightField"
                     softwareVersion="4.2.1.0"
                     softwareCompany="Princeton Instruments">
            <FrameCombination method="Sum"
                              framesCombined="2" />
      </DataModified>
   </DataHistory>
</DataHistories>
```

Figure 11. Example 7: DataHistories Element

4.) The **DataHistories** element is optional.
5.) If **DataHistories** exists, there will be one child **DataHistory** element representing a timeline of changes to the image data.

6.) A **DataHistory** timeline contains an **Origin** element detailing the initial collection of the image data.
   a. This element contains **creator** and **created** attributes describing who captured the data and the date the data was taken, respectively.
      - The **creator** attribute names the user who acquired the data.
      - The **created** attribute notes the day and time of acquisition in w3c format.
      - Additionally, the optional **software**, **softwareVersion** and **softwareCompany** attributes provide may details about the acquisition software used.
   b. The child element of **Origin** provides additional details related to the acquisition. This element is left open-ended for customization and is not defined by SPE 3.0. For LightField, the child element is an **Experiment** element in its own namespace. For more information, refer to the Experiment XML specification.

7.) A **DataHistory** timeline represents modification to the image data with **DataModified** child elements.
   a. This element contains **user** and **date** attributes describing who captured the data and the date the data was taken, respectively.
      - The **user** attribute names the user who acquired the data.
      - The **date** attribute notes the day and time of acquisition in w3c format.
      - Additionally, the optional **software**, **softwareVersion** and **softwareCompany** attributes provide may details about the acquisition software used.
   b. The child elements of **DataModified** provide additional details related to the post-processing of the image data.
      - For **BackgroundCorrection**, the **reference** attribute reports the name and location of the file used in the correction.
      - For **BlemishCorrection**, the **definition** attribute reports the name and location of the file used in the correction.
      - For **CosmicRayCorrection**, the **method** and **kernelSize** attributes report the filter and kernel size used.
         1. The **method** attribute value is either **MedianFilter** or **DespeckleFilter**
         2. The **kernelSize** attribute value is either **3**, **5**, or **7** to indicate that a 3x3, 5x5, or 7x7 matrix was used.
      - For **FlatfieldCorrection**, the **reference** attribute reports the name and location of the file used in the correction.
      - For **FrameCombination**, the **method** and framesCombined attributes report the method used and the number of frames that were combined to create a frame.
         1. The **method** value is either **Sum** or **Average**.
         2. The **framesCombined** value is number of frames combined into a single frame.
      - For **OrientationCorrection**, the **method** attribute reports the reports the rotation applied to the image data.
         1. The **method** value is one or more of the following (comma delimited):
            a. **FlippedHorizontally** – the sensor is reflected from left to right of the normal orientation.
            b. **FlippedVertically** – the sensor is reflected from top to bottom the normal orientation.
            c. **RotatedClockwise** – the sensor is rotated clockwise in relation to the normal orientation.
         2. Rotation is always applied after any flips. Another point of view is that rotation rotates the axes of symmetry for reflection as well.

- For **SoftwareBinning**, the **format** attribute reports the X and Y binning values used as in a **format** value example of **"XYBinningValues">2,4** where 2 is the amount of horizontal binning and 4 is the vertical).
- The **DataExtraction** attributes define the origin and the size of the region of interest that was extracted.
  1. The **DataExtraction** attributes are **x**, **width**, **y**, and **height**.
     a. **x** and **y** define the origin (the zero-based top-left corner of the region).
     b. **width** and **height** describe the size in pixels.
  2. **DataSelection** is a child element of **DataExtraction**. The attributes **frameStart** and **frameEnd** define which frames will be included in the extraction. If there are five frames in the data and **frameStart** = **1** and **frameEnd** = **3**, then frames 1, 2, and 3 of the data will be included.
  3. **RegionOfInterest** is a child element of **DataSelection**. Its **x**, **width**, **xBinning**, **y**, **height**, and **yBinning** attributes describe the size of the source data for the extraction, including any binning.
     a. The **x** and **y** attributes describe the top-left corner of the region (zero-based).
     b. The **height** and **width** attributes describe the size of the region in pixels.
     c. The **xBinning** and **yBinning** attributes describe the combination of columns and rows for the region.
- The **CrossSection** element and its child elements describe the type and method used for the process as well as the location, and dimensions of the region used to generate the cross section. It also describes the origin, dimensions, and binning for the source data used.
  1. The **CrossSection** attributes are **type**, **method**, **x**, **width**, **y**, and **height** .
     c. **type** has the value of **Horizontal**, **Vertical**, or **Frame**.
     d. **method** has the value of **Sum** or **Average**.
     e. **x** and **y** define the origin (the zero-based top-left corner of the region).
     f. **width** and **height** describe the size in pixels.
  2. **DataSelection** is a child element of **CrossSection**. The attributes **frameStart** and **frameEnd** define which frames will be included in the cross section. If there are five frames in the data and **frameStart** = **1** and **frameEnd** = **3**, then frames 1, 2, and 3 of the data will be included.
  3. **RegionOfInterest** is a child element of **DataSelection**. Its **x**, **width**, **xBinning**, **y**, **height**, and **yBinning** attributes describe the size of the source data for the cross section, including any binning.
     a. **x** and **y** describe the top-left corner of the region of interest (zero-based).
     b. **height** and **width** attributes describe the size of the region in pixels.
     c. **xBinning** and **yBinning** attributes describe the combination of columns and rows for the region.

This page intentionally left blank.

Chapter 6
Covering Miscellaneous Information

The final and optional section of the SPE 3.0 footer is the **GeneralInformation** element which describes basic information about the creator and the dates the file was created and modified.

```
<GeneralInformation>
        <FileInformation creator="jjones"
                         created="2012-01-18T11:00:45.3942594-05:00"
                         lastModified="2012-04-03T14:45:29.2781542-04:00" />
        <Notes xml:space="preserve">Notes entered on the File Information|Notes
                         tab will appear here.</Notes>
</GeneralInformation>
```

Figure 12. Example 8: GeneralInformation Element

The optional **FileInformation** element contains **creator** and **created** attributes describing who captured the data and the date the data was taken, respectively. It also contains the **lastModified** attribute which reports the last time the file contents were modified. If the file contents have never been modified, **created** and **lastModified** will be identical. In the example in Figure 12, the file was created on January 18, 2012 and subsequently modified on April 3, 2012.

- The **creator** attribute names the user who acquired the data.
- The **created** attribute notes the day and time of acquisition in w3c format.
- The **lastModified** attribute must in a round-trip-friendly w3c date time format.

The optional **Notes** element reports the user-entered text from the **File Information|Notes** tab (accessed in the LightField Data workspace).

This page intentionally left blank.

Appendix A
SPE 2.x Header with Changes

Introduction

The tables that follow describe the 2.X header (with changes) and are provided as a reference. The SPE 3.0 header only requires entries in Offset Locations **678** and **1992** (highlighted in blue). However, for an SPE 3.0 data file to be read by WinX (WinSpec, for example), the locations highlighted in yellow must **also** be included in the SPE 3.0 header.

Start of Header Information (0 - 2996)

Binary Type	Name	Offset	Description
16s	ControllerVersion	0	Hardware Version
16s	LogicOutput	2	Definition of Output BNC
16u	AmpHiCapLowNoise	4	Amp Switching Mode
16u	xDimDet	6	Detector x dimension of chip.
16s	mode	8	timing mode
32f	exp_sec	10	alternative exposure, in sec.
16s	VChipXdim	14	Virtual Chip X dim
16s	VChipYdim	16	Virtual Chip Y dim
16u	yDimDet	18	y dimension of CCD or detector.
8s	date[DATEMAX]	20	date
16s	VirtualChipFlag	30	On/Off
8s	Spare_1[2]	32	
16s	noscan	34	Old number of scans - should always be -1
32f	DetTemperature	36	Detector Temperature Set
16s	DetType	40	CCD/DiodeArray type
16u	xdim	42	actual # of pixels on x axis
16s	stdiode	44	trigger diode
32f	DelayTime	46	Used with Async Mode
16u	ShutterControl	50	Normal, Disabled Open, Disabled Closed
16s	AbsorbLive	52	On/Off
16u	AbsorbMode	54	Reference Strip or File
16s	CanDoVirtualChipFlag	56	T/F Cont/Chip able to do Virtual Chip
16s	ThresholdMinLive	58	On/Off
32f	ThresholdMinVal	60	Threshold Minimum Value
16s	ThresholdMaxLive	64	On/Off
32f	ThresholdMaxVal	66	Threshold Maximum Value
16s	SpecAutoSpectroMode	70	T/F Spectrograph Used

Binary Type	Name	Offset	Description
32f	SpecCenterWlNm	72	Center Wavelength in Nm
16s	SpecGlueFlag	76	T/F File is Glued
32f	SpecGlueStartWlNm	78	Starting Wavelength in Nm
32f	SpecGlueEndWlNm	82	Starting Wavelength in Nm
32f	SpecGlueMinOvrlpNm	86	Minimum Overlap in Nm
32f	SpecGlueFinalResNm	90	Final Resolution in Nm
16s	PulserType	94	0=None, PG200=1, PTG=2, DG535=3
16s	CustomChipFlag	96	T/F Custom Chip Used
16s	XPrePixels	98	Pre Pixels in X direction
16s	XPostPixels	100	Post Pixels in X direction
16s	YPrePixels	102	Pre Pixels in Y direction
16s	YPostPixels	104	Post Pixels in Y direction
16s	asynen	106	asynchron enable flag 0 = off
16s	datatype	108	experiment datatype 0 = 32f (4 bytes) 1 = 32s (4 bytes) 2 = 16s (2 bytes) 3 = 16u (2 bytes) 8 = 32u (4 bytes)
16s	PulserMode	110	Repetitive/Sequential
16u	PulserOnChipAccums	112	Num PTG On-Chip Accums
32u	PulserRepeatExp	114	Num Exp Repeats (Pulser SW Accum)
32f	PulseRepWidth	118	Width Value for Repetitive pulse (usec)
32f	PulseRepDelay	122	Width Value for Repetitive pulse (usec)
32f	PulseSeqStartWidth	126	Start Width for Sequential pulse (usec)
32f	PulseSeqEndWidth	130	End Width for Sequential pulse (usec)
32f	PulseSeqStartDelay	134	Start Delay for Sequential pulse (usec)
32f	PulseSeqEndDelay	138	End Delay for Sequential pulse (usec)
16s	PulseSeqIncMode	142	Increments: 1=Fixed, 2=Exponential
16s	PImaxUsed	144	PI-Max type controller flag
16s	PImaxMode	146	PI-Max mode
16s	PImaxGain	148	PI-Max Gain
16s	BackGrndApplied	150	1 if background subtraction done
16s	PImax2nsBrdUsed	152	T/F PI-Max 2ns Board Used
16u	minblk	154	min. # of strips per skips
16u	numminblk	156	# of min-blocks before geo skps
16s	SpecMirrorLocation[2]	158	Spectro Mirror Location, 0=Not Present
16s	SpecSlitLocation[4]	162	Spectro Slit Location, 0=Not Present

Binary Type	Name	Offset	Description
16s	CustomTimingFlag	170	T/F Custom Timing Used
8s	ExperimentTimeLocal[TIMEMAX]	172	Experiment Local Time as hhmmss\0
8s	ExperimentTimeUTC[TIMEMAX]	179	Experiment UTC Time as hhmmss\0
16s	ExposUnits	186	User Units for Exposure
16u	ADCoffset	188	ADC offset
16u	ADCrate	190	ADC rate
16u	ADCtype	192	ADC type
16u	ADCresolution	194	ADC resolution
16u	ADCbitAdjust	196	ADC bit adjust
16u	gain	198	gain
8s	Comments[5][COMMENTMAX]	200	File Comments
16u	geometric	600	geometric ops: rotate 0x01,reverse 0x02, flip 0x04
8s	xlabel[LABELMAX]	602	intensity display string
16u	cleans	618	cleans
16u	NumSkpPerCln	620	number of skips per clean.
16s	SpecMirrorPos[2]	622	Spectrograph Mirror Positions
32f	SpecSlitPos[4]	626	Spectrograph Slit Positions
16s	AutoCleansActive	642	T/F
16s	UseContCleansInst	644	T/F
16s	AbsorbStripNum	646	Absorbance Strip Number
16s	SpecSlitPosUnits	648	Spectrograph Slit Position Units
32f	SpecGrooves	650	Spectrograph Grating Grooves
16s	srccmp	654	number of source comp.diodes
16u	ydim	656	y dimension of raw data.
16s	scramble	658	0=scrambled,1=unscrambled
16s	ContinuousCleansFlag	660	T/F Continuous Cleans Timing Option
16s	ExternalTriggerFlag	662	T/F External Trigger Timing Option
32s	lnoscan	664	Number of scans (Early WinX)
32s	lavgexp	668	Number of Accumulations
32f	ReadoutTime	672	Experiment readout time
16s	TriggeredModeFlag	676	T/F Triggered Timing Option
64u	XML Offset	678	Starting location of the XML footer
8s	sw_version[FILEVERMAX]	688	Version of SW creating this file
16s	type	704	1 = new120 (Type II) 2 = old120 (Type I) 3 = ST130 4 = ST121 5 = ST138 6 = DC131 (PentaMax) 7 = ST133 (MicroMax/SpectroMax) 8 = ST135 (GPIB)

Binary Type	Name	Offset	Description
			9 = VICCD 10 = ST116 (GPIB) 11 = OMA3 (GPIB) 12 = OMA4
16s	flatFieldApplied	706	1 if flat field was applied
8s	Spare_3[16]	708	
16s	kin_trig_mode	724	Kinetics Trigger Mode
8s	dlabel[LABELMAX]	726	Data label.
8s	Spare_4[436]	742	
8s	PulseFileName[HDRNAMEMAX]	1178	Name of Pulser File with Pulse Widths/Delays (for Z-Slice)
8s	AbsorbFileName[HDRNAMEMAX]	1298	Name of Absorbance File (if File Mode)
32u	NumExpRepeats	1418	Number of Times experiment repeated
32u	NumExpAccums	1422	Number of Time experiment accumulated
16s	YT_Flag	1426	Set to 1 if this file contains YT data
32f	clkspd_us	1428	Vert Clock Speed in micro-sec
16s	HWaccumFlag	1432	set to 1 if accum done by Hardware.
16s	StoreSync	1434	set to 1 if store sync used
16s	BlemishApplied	1436	set to 1 if blemish removal applied
16s	CosmicApplied	1438	set to 1 if cosmic ray removal applied
16s	CosmicType	1440	if cosmic ray applied, this is type
32f	CosmicThreshold	1442	Threshold of cosmic ray removal.
32s	NumFrames	1446	number of frames in file.
32f	MaxIntensity	1450	max intensity of data (future)
32f	MinIntensity	1454	min intensity of data future)
8s	ylabel[LABELMAX]	1458	y axis label.
16u	ShutterType	1474	shutter type.
32f	shutterComp	1476	shutter compensation time.
16u	readoutMode	1480	readout mode, full,kinetics, etc
16u	WindowSize	1482	window size for kinetics only.
16u	clkspd	1484	clock speed for kinetics & frame transfer
16u	interface_type	1486	computer interface (isa-taxi, pci, eisa, etc.)
16s	NumROIsInExperiment	1488	May be more than the 10 allowed in this header (if 0, assume 1)
8s	Spare_5[16]	1490	
16u	controllerNum	1506	if multiple controller system will have controller number data came from. This is a future item.
16u	SWmade	1508	Which software package created this file

Binary Type	Name	Offset	Description
16s	NumROI	1510	number of ROIs used. if 0 assume 1.
Struct ROIinfo{			
16u	startx		left x start value.
16u	endx		right x value.
16u	groupx		amount x is binned/grouped in hw.
16u	starty		top y start value.
16u	endy		bottom y value.
16u	groupy		amount y is binned/grouped in hw.
} ROIinfoblk[ROIMAX]			ROI Starting Offsets
		1512	ROI 1
		1524	ROI 2
		1536	ROI 3
		1548	ROI 4
		1560	ROI 5
		1572	ROI 6
		1584	ROI 7
		1596	ROI 8
		1608	ROI 9
		1620	ROI 10
8s	FlatField[HDRNAMEMAX]	1632	Flat field file name.
8s	background[HDRNAMEMAX]	1752	background sub. file name.
8s	blemish[HDRNAMEMAX]	1872	blemish file name.
32f	file_header_ver	1992	version of this file header **(3.0)**
8s	YT_Info[1000]	1996-2995	Reserved for YT information
32s	WinView_id	2996	== 0x01234567L if file created by WinX

Calibration Structures

There are three structures for the calibrations

- The Area Inside the Calibration Structure (below) is repeated two times.

```
xcalibration,        /* 3000 – 3488 x axis calibration*/
ycalibration,        /* 3489 – 3977 y axis calibration*/
```

Start of X Calibration Structure (3000 - 3488)

Binary Type	Name	Offset	Description
64f	offset	3000	offset for absolute data scaling
64f	factor	3008	factor for absolute data scaling
8s	current_unit	3016	selected scaling unit
8s	reserved1	3017	reserved
8s	string[40]	3018	special string for scaling

Binary Type	Name	Offset	Description
8s	reserved2[40]	3058	reserved
8s	calib_valid	3098	flag if calibration is valid
8s	input_unit	3099	current input units for "calib_value"
8s	polynom_unit	3100	linear UNIT and used in the "polynom_coeff"
8s	polynom_order	3101	ORDER of calibration POLYNOM
8s	calib_count	3102	valid calibration data pairs
64f	pixel_position[10]	3103	pixel pos. of calibration data
64f	calib_value[10]	3183	calibration VALUE at above pos
64f	polynom_coeff[6]	3263	polynom COEFFICIENTS
64f	laser_position	3311	laser wavenumber for relative WN
8s	reserved3	3319	reserved
8u	new_calib_flag	3320	If set to 200, valid label below
8s	calib_label[81]	3321	Calibration label (NULL term'd)
8s	expansion[87]	3402	Calibration Expansion area

Start of Y Calibration Structure (3489 - 3977)

Binary Type	Name	Offset	Description
64f	offset	3489	offset for absolute data scaling
64f	factor	3497	factor for absolute data scaling
8s	current_unit	3505	selected scaling unit
8s	reserved1	3506	reserved
8s	string[40]	3507	special string for scaling
8s	reserved2[40]	3547	reserved
8s	calib_valid	3587	flag if calibration is valid
8s	input_unit	3588	current input units for "calib_value"
8s	polynom_unit	3589	linear UNIT and used in the "polynom_coeff"
8s	polynom_order	3590	ORDER of calibration POLYNOM
8s	calib_count	3591	valid calibration data pairs
64f	pixel_position[10]	3592	pixel pos. of calibration data
64f	calib_value[10]	3672	calibration VALUE at above pos
64f	polynom_coeff[6]	3752	polynom COEFFICIENTS
64f	laser_position	3800	laser wavenumber for relative WN
8s	reserved3	3808	reserved
8u	new_calib_flag	3809	If set to 200, valid label below
8s	calib_label[81]	3810	Calibration label (NULL term'd)
8s	expansion[87]	3891	Calibration Expansion area

End of Calibration Structures (3978-4098)

Binary Type	Name	Offset	Description
8s	Istring[40]	3978	special intensity scaling string
8s	Spare_6[25]	4018	
8u	SpecType	4043	spectrometer type (acton, spex, etc.)
8u	SpecModel	4044	spectrometer model (type dependent)
8u	PulseBurstUsed	4045	pulser burst mode on/off
32u	PulseBurstCount	4046	pulser triggers per burst
64f	PulseBurstPeriod	4050	pulser burst period (in usec)
8u	PulseBracketUsed	4058	pulser bracket pulsing on/off
8u	PulseBracketType	4059	pulser bracket pulsing type
64f	PulseTimeConstFast	4060	pulser slow exponential time constant (in usec)
64f	PulseAmplitudeFast	4068	pulser fast exponential amplitude constant
64f	PulseTimeConstSlow	4076	pulser slow exponential time constant (in usec)
64f	PulseAmplitudeSlow	4084	pulser slow exponential amplitude constant
16s	AnalogGain;	4092	analog gain
16s	AvGainUsed	4094	avalanche gain was used
16s	AvGain	4096	avalanche gain value
16s	lastvalue	4098	Always the LAST value in the header

This page intentionally left blank.

Index

T-Z

Princeton Instruments

www.princetoninstruments.com

TOLL-FREE +1.877.474.2286 | PHONE +1.609.587.9797