rapidlasso GmbH - fast tools to catch reality
contact: Martin Isenburg
cell: 0049-176-3462-3245
email: martin@rapidlasso.com
web: http://rapidlasso.com

To Whom It May Concern:

Below is our bid for the "ASPRS LAS Validation Suite" in response to the request for proposals.

1. Cost

USD 24000 (in words: twenty-four thousand US dollars)

2. Technical

We propose to create an open source validation tool using the open source LASlibrary API which was designed particularly for bidding on this request for proposals. The LASlibrary package is an unrestricted open source (LGPL 2.1) API to read LAS (optionally also LAZ) files. It is a *completely re-engineered subset* of the LASlib API of LAStools that has been drastically simplified to better suit the being-light-weight-requirement of the ASPRS' request for proposals.
These are the advantages of implementing the validation tool in open source using the new light-weight open source LASlibrary API:

1. The source code of LASlibrary has a proven track record. It is in large parts derived from the robust LASlib of LAStools which has seen wide use over many years.

2. Because the aforementioned LASlibrary is almost complete we are able to demonstrate our commitment to the task ahead by providing a ready-to-run prototype of the proposed validation tool. This demo version is available here: http://github.com/LASvalidator/

3. We can soon start to crowd-test and community-improve the functionality of the proposed validation tool. The complexity of the task at hand should not be underestimated. In light of the many "flavors" of LAS files that we have seen it will be difficult for any one party to cover all possible scenarious in which a LAS file can be valid or corrupt. By opening up the tool development process *early*, the completeness and correctness of the proposed validation checks can be tested in a holistic manner with community support. This will also shorten the total time required until project completion.

4. The proposed openness in the development process with opportunity for early feedback will enhance the credibility of the LAS validation tool and result in speedy acceptance by the LiDAR community. There have been worries in the community that the validation tool could be used as some form of "lock-out" mechanism. The open approach proposed here will dispel any concerns of there being hidden agendas.

5. The prototype code is designed in a modular fashion such that it would be easily possible to add the needed functionality to fix minor issues that LAS files may have such as fitting a proper bounding box, repairing counters, correcting the creation date, adjusting a bit in the global encoding field, etc …

6. Finally, the use of open source LPGL 2.1 for the project allows seamless inclusion of the LASzip compressor into the LASlibrary so that LAS files – assuming that this is a desired feature – could also be verified directly in their lossless compressed form as LAZ files.

An (incomplete) prototype and example XML output is available: http://github.com/LASvalidator/.

rapidlasso GmbH - fast tools to catch reality
contact: Martin Isenburg
cell: 0049-176-3462-3245
email: martin@rapidlasso.com
web: http://rapidlasso.com

Technical details:

As already mentioned, the entire LAS file handling will be done using the LASlibrary API that is a subset of LASlib from LAStools with an unrestricted LGPL 2.1 open source license. In Appendix C we list example header files from the LASlibrary API, namely that of LASreader - the class doing the actual reading of the LAS files – and that of LASheader – the class storing all header information including VLRs and EVLRs.

The required validations checks are fairly straight forward to implement because the LASlibrary API already takes care of all the low-level details. Hence, the main issues that remain are the exact command-line parameters. the behavior in case of multiple files with the same name whose summaries are written to the same directory, the input and output XML formats, a few additional items that might be worth checking, and – most importantly - the CRS handling.

Command line parameters:

input

`-i lidar.las`	: one LAS file
`-i lidar1.las lidar2.las`	: two (or more) LAS files
`-i LDR*.las`	: all LAS files that match the wildcard "LDR*.las"
`-i strips\LDR*.las`	: all LAS files in folder "strips" that match the wildcard
`-lof list_of_files.txt`	: all LAS files listed in an ASCII file
`-idir flight_lines`	: all LAS files in a directory
`-idir flight1 flight2`	: all LAS files in two (or more) directories
`-irdir flight_lines`	: all LAS files in a directory and its subdirectories
`-irdir flight1 flight2`	: all LAS files in two directories and their subdirectories
`-ixml input.xml`	: all LAS files specified in the XML file

output

`-o summary.xml`	: all XML reports go into one summary file
`-oxml`	: each lidar.las file gets a lidar_LVE.xml summary in its directory
`-odir reports`	: same as previous but puts all summaries into one directory
`-ordir reports`	: same as previous but puts them into recursive directories if the input was recursive as well, otherwise identical to '-odir'

Multiple files written to the same directory:

This is one important thing needs to be discussed: In case the input consists of multiple input files from different directories – which can happen in several different ways – and the output is to go into one directory then overwriting of summaries would occur. How to distinguish between just written summaries and pre-existing files? How to best resolve this issue will be addressed during the open development phase.

If two instances of the tool run at the same time they will not clash when reading the same LAS file but when trying to write the summary at the same time into the same directory. The tool that is second to write can either abort altogether or only skip this LAS file's summary. During the open development phase we will decide which of the two is the better solution for this scenario.

input and output XML format

The proposed input and the output XML formats are presented in Appendix A and Appendix B respectively. They are (will be) implemented with a clean and simple XMLwriter (XMLreader).

Other potential checks

In theory points of a LAS file are neither allowed to have return counts or number of returns of given pulse values of zero, nor is the return count allowed to be larger than the number of returns of given pulse – both are violations of the specification that we have seen occurring frequently.

Furthermore – if one wanted to be strict – then the highest return count and the highest number of returns of given pulse value for any point for LAS specifications 1.0 to 1.3 would be 5. Many files we have seen make full use of the three bits and allow return counts that go up to 7. One could deal with this not by failing these files but by emitting a warning for all files where this occurs.

The RGBI values as well as the intensities are supposed to make use of the full 16 bit resolution of their respective fields. It would easily be possible to add a check whether their actual range across all points in a LAS file is limited to the bottom 8 bits. From this we could conclude that the scaling is not in accordance with the LAS specification and a warning (?) could be emitted.

Some LAS files use, for example, point type 3 (instead of point type 1) despite the RGB values of every point in the file being zero. Similarly, we have come across quite a number of LAS files using point type 1 (instead of point type 0) despite the GPS time stamp of every point in the file being zero. It would seem reasonable to emit a warning for such cases.

These are just a few possibilities for additional validation checks that could be easily implemented into the demo version of our prototype validation tool hosted at http://github.com/LASvalidator/.

Presence and Validity of Coordinate Reference System (CRS)

The Coordinate Reference System (CRS) checking will be implemented by utilizing two open source libraries: GeoTIFF and GDAL. The GeoTIFF library will be used to convert the GeoTIFF tags into OGC WKT strings and these OGC WKT strings are then always checked with GDAL for validity. Valid GeoTIFF tags must be present in any LAS file containing point data formats 0 to 5 for the CRS test to pass. A valid OGC WKT string must be present in any LAS file containing point data formats 6 and higher for the CRS test to pass. GeoTIFF tags may also be present in addition to the OGC WKT string in LAS file containing point data formats 6 and higher. In this case they must not only be valid but also describe the exact same Coordinate Reference System as the OGC WKT string. After bringing them both into the same representation one may think that a simple string compare is sufficient for checking this. However, there is the issue with vertical datums that have poor support in GeoTIFF tags which was one of the main reasons for the LAS Working Group to switch to OGC WKT strings. I suggest to issue only a warning (and not a fail) when vertical datums are missing in the GeoTIFF tags or - to be slightly more strict - to issue a fail only when the vertical datum is one of the few well supported ones that could have been easily added (e.g. NAVD88, WGS84, ...). Should a more complex vertical datum GeoTIFF be required we will also include the proj4 library and obtain input from experts such as Frank Wanderdam on how to do this in the most compatible manner.

While GeoTIFF and GDAL are arguably the most comprehensive and robust option for checking the correctness of the Coordinate Reference System, it means to make the validation tool

dependent on two (or even three) other open source libraries - which is not in harmony with the ASPRS' request for simplicity in compilation of the source code. One possible solution would be to extract only those parts from the GeoTIFF and GDAL libraries that are relevant for checking the CRS requirements and then provide those as a small open source CRSlibrary library that is part of the validation software source distribution in a similar fashion as LASlibrary. The existence of such a CRSlibrary could - all by itself – turn into a useful measure to improve the overall quality of future LAS files as it would provide the LiDAR community with a simpler package for the admittedly tricky task of CRS handling. We would be pursue whichever option (use existing GeoTIFF and GDAL libs or extract new CRSlibrary) is considered more worthwhile by the ASPRS LAS committee and the open source community that has been using and is in the know about GeoTIFF, GDAL, and proj4.

In summary, we propose to implement the LAS validation tool through an open process that allows early community feedback. We will produce new and suitably simplified open source libraries that we derive from robust and successful open source packages which are accepted gold standards and already widely used. Providing an "official correctness check" for LAS files and especially that of the Coordinate Reference System is likely to spark emotionally charged discussions. We doubt that any one party will - on their own and behind closed doors - be able to develop a comprehensive LAS validation software that will find speedy acceptance by an already "slightly suspicious" LiDAR community. The open approach we propose here will disarm any notions that this software might be used as a "lock-out" mechanism, will develop the desired product in a rapid and transparent manner, and will produce a validation tool that will be quickly available for professional use and also quick in gaining acceptance by the LiDAR community.

Regards,

Dr. Martin Isenburg

(scientist and owner at rapidlasso GmbH)

3. References

Amar Nayegandhi, *Dewberry*, anayegandhi@dewberry.com
Christopher Hopkinson, *University of Lethbridge*, c.hopkinson@uleth.ca
David Finnegan, *USACE*, david.finnegan@usace.army.mil
Jeff Fagermann, *Scanview*, jeff.fagerman@gmail.com
Bob McGaughey, *USDA FUSION*, mcgoy@u.washington.edu
Dale Lutz, *Safe Software*, Dale.Lutz@safe.com
Doug Newcomb, *US Fish and Wildlife Services*, doug_newcomb@fws.gov
Mike Childs, *Global Mapper*, mikec@bluemarblegeo.com
Jason Stoker, *USGS*, jstoker@usgs.gov
Frank Wanderdam, *Google*, warmerdam@pobox.com

rapidlasso GmbH - fast tools to catch reality
contact: Martin Isenburg
cell: 0049-176-3462-3245
email: martin@rapidlasso.com
web: http://rapidlasso.com

Radu Rusu, *Point Cloud Library*, rusu@openperception.org
Michelle Quirk, *NGA*, pal@math.utexas.edu
Kirk Waters, *NOAA*, kirk.waters@noaa.gov
Ted Knaak, *Certainty3D*, ted.knaak@certainty3d.com
Michael Gerlek, *Flaxen Consulting*, mpg@flaxen.com
Mauricio Terneus, *Certainty3D*, mauricio.terneus@certainty3d.com
Ananda Fowler, *RIEGL*, afowler@rieglusa.com
Christopher Crosby, *OpenTopography*, ccrosby@sdsc.edu
Tim Loesch, *DNR Minnesota*, tim.loesch@state.mn.us

Appendix A:

Here an illustrative example of how the proposed XML input will look like. It consists of up to three different sections (in practice probably only one) that list one or multiple full <path> to LAS files bracketed by <files>, directories full of LAS files bracketed by <dirs>, or directories that are to be recursively browsed for LAS files bracketed by <rdirs> as shown in the following XML example:

```
<?xml version="1.0" encoding="UTF-8"?>
<LASinput>
  <files>
    <path>C:\lasvalidator\lasvalidate\data\fusa.las</path>
    <path>C:\lasvalidator\lasvalidate\data\lake.las</path>
    <path>C:\lasvalidator\lasvalidate\data\test.las</path>
    <path>C:\lasvalidator\lasvalidate\data\TO_core_last_zoom.las</path>
  </files>
  <dirs>
    <path>C:\lasvalidator\lasvalidate\flight1\</path>
    <path>C:\lasvalidator\lasvalidate\flight2\</path>
  </dirs>
  <rdirs>
    <path>C:\lasvalidator\lasvalidate\</path>
  </rdirs>
</LASinput>
```

Appendix B:

Below a few illustrative examples on how the proposed XML output will look like. For each LAS file there will be one <report> consisting of a description of the <file> of a single <summary> of the test result and optional <details> that provide more information about the reasons why the file failed or why there were warnings.

Example 1 – Single file "pass"

```
<?xml version="1.0" encoding="UTF-8"?>
<LASvalidator>
  <report>
    <file>
      <name>fusa.las</name>
      <path>C:\lasvalidator\lasvalidate\data\fusa.las</path>
      <version_major>1</version_major>
      <version_minor>1</version_minor>
      <point_data_format>1</point_data_format>
      <CRS>not implemented (yet)</CRS>
```


```
      </file>
      <summary>
        pass
      </summary>
    </report>
    <total>
      pass
      <details>
        <pass>1</pass>
        <warning>0</warning>
        <fail>0</fail>
      </details>
    </total>
  </LASvalidator>
```

Example 2 – Single file "fail"

```
  <LASvalidator>
    <report>
      <file>
        <name>TO_core_last_zoom.las</name>
        <path>C:\lasvalidator\lasvalidate\data\TO_core_last_zoom.las</path>
        <version_major>1</version_major>
        <version_minor>1</version_minor>
        <point_data_format>1</point_data_format>
        <CRS>not implemented (yet)</CRS>
      </file>
      <summary>
        fail
      </summary>
      <details>
        <warning>
          <variable>system identifier</variable>
          <note>empty string. first character is '\0'</note>
        </warning>
        <fail>
          <variable>file creation day</variable>
          <note>not set</note>
        </fail>
        <fail>
          <variable>file creation year</variable>
          <note>not set</note>
        </fail>
        <fail>
          <variable>CRS</variable>
          <note>file does not specify a Coordinate Reference System</note>
        </fail>
      </details>
    </report>
    <total>
      fail
      <details>
        <pass>0</pass>
        <warning>0</warning>
        <fail>1</fail>
      </details>
    </total>
```


```
</LASvalidator>
```

Example 3 – Multiple file report

```
<?xml version="1.0" encoding="UTF-8"?>
<LASvalidator>
  <report>
    <file>
      <name>fusa.las</name>
      <path>C:\lasvalidator\lasvalidate\data\fusa.las</path>
      <version_major>1</version_major>
      <version_minor>1</version_minor>
      <point_data_format>1</point_data_format>
      <CRS>not implemented (yet)</CRS>
    </file>
    <summary>
      pass
    </summary>
  </report>
  <report>
    <file>
      <name>lake.las</name>
      <path>C:\lasvalidator\lasvalidate\data\lake.las</path>
      <version_major>1</version_major>
      <version_minor>1</version_minor>
      <point_data_format>1</point_data_format>
      <CRS>not implemented (yet)</CRS>
    </file>
    <summary>
      fail
    </summary>
    <details>
      <fail>
        <variable>CRS</variable>
        <note>file does not specify a Coordinate Reference System</note>
      </fail>
    </details>
  </report>
  <report>
    <file>
      <name>test.las</name>
      <path>C:\lasvalidator\lasvalidate\data\test.las</path>
      <version_major>1</version_major>
      <version_minor>1</version_minor>
      <point_data_format>0</point_data_format>
      <CRS>not implemented (yet)</CRS>
    </file>
    <summary>
      fail
    </summary>
    <details>
      <warning>
        <variable>system identifier</variable>
        <note>empty string. first character is '\0'</note>
      </warning>
      <fail>
        <variable>file creation year</variable>
```


```
        <note>should be between 1990 and 2013 and not 1</note>
      </fail>
    </details>
  </report>
  <report>
    <file>
      <name>TO_core_last_zoom.las</name>
      <path>C:\lasvalidator\lasvalidate\data\TO_core_last_zoom.las</path>
      <version_major>1</version_major>
      <version_minor>1</version_minor>
      <point_data_format>1</point_data_format>
      <CRS>not implemented (yet)</CRS>
    </file>
    <summary>
      fail
    </summary>
    <details>
      <warning>
        <variable>system identifier</variable>
        <note>empty string. first character is '\0'</note>
      </warning>
      <fail>
        <variable>file creation day</variable>
        <note>not set</note>
      </fail>
      <fail>
        <variable>file creation year</variable>
        <note>not set</note>
      </fail>
      <fail>
        <variable>CRS</variable>
        <note>file does not specify a Coordinate Reference System</note>
      </fail>
    </details>
  </report>
  <total>
    fail
    <details>
      <pass>1</pass>
      <warning>0</warning>
      <fail>3</fail>
    </details>
  </total>
</LASvalidator>
```

Appendix C:

Below are a few code samples to give those evaluating out bid a chance to see the coding style that they can expect the open source validator tool to follow. For more example you can see the few example header files that we have made available at: http://github.com/LASvalidator/.

Example 1 – class LASreader

```
class LASreader
{
public:
```


```
    LASheader header;
    LASpoint point;

    I64 npoints;
    I64 p_count;

    BOOL open(const char* file_name, U32 io_buffer_size=65536);
    BOOL open(FILE* file);
    BOOL open(istream& stream);

    BOOL is_compressed() const;

    inline F64 get_min_x() const { return header.min_x; };
    inline F64 get_min_y() const { return header.min_y; };
    inline F64 get_min_z() const { return header.min_z; };

    inline F64 get_max_x() const { return header.max_x; };
    inline F64 get_max_y() const { return header.max_y; };
    inline F64 get_max_z() const { return header.max_z; };

    BOOL seek(const I64 p_index);
    BOOL read_point();

    inline F64 get_x() const { return point.get_x(); };
    inline F64 get_y() const { return point.get_y(); };
    inline F64 get_z() const { return point.get_z(); };

    void close(BOOL close_stream=TRUE);

    LASreader();
    ~LASreader();

  private:
    BOOL open(ByteStreamIn* stream);
    FILE* file;
    ByteStreamIn* stream;
    LASreadPoint* reader;
  };
```

Example 2 – class LASheader

```
  class LASheader : public LASquantizer, public LASattributer
  {
  public:
    CHAR file_signature[4];
    U16 file_source_id;
    U16 global_encoding;
    U32 project_ID_GUID_data_1;
    U16 project_ID_GUID_data_2;
    U16 project_ID_GUID_data_3;
    CHAR project_ID_GUID_data_4[8];
    U8 version_major;
    U8 version_minor;
    CHAR system_identifier[32];
    CHAR generating_software[32];
    U16 file_creation_day;
    U16 file_creation_year;
```


```
    U16 header_size;
    U32 offset_to_point_data;
    U32 number_of_variable_length_records;
    U8 point_data_format;
    U16 point_data_record_length;
    U32 legacy_number_of_point_records;
    U32 legacy_number_of_points_by_return[5];
    F64 max_x;
    F64 min_x;
    F64 max_y;
    F64 min_y;
    F64 max_z;
    F64 min_z;

    // LAS 1.3 and higher only
    U64 start_of_waveform_data_packet_record;

    // LAS 1.4 and higer only
    U64 start_of_first_extended_variable_length_record;
    U32 number_of_extended_variable_length_records;
    U64 number_of_point_records;
    U64 number_of_points_by_return[15];

    U32 user_data_in_header_size;
    U8* user_data_in_header;

    LASvlr* vlrs;
    LASevlr* evlrs;
    LASvlr_geo_keys* vlr_geo_keys;
    LASvlr_key_entry* vlr_geo_key_entries;
    F64* vlr_geo_double_params;
    CHAR* vlr_geo_ascii_params;
    CHAR* vlr_geo_ogc_wkt;
    LASvlr_classification* vlr_classification;
    LASvlr_wave_packet_descr** vlr_wave_packet_descr;

    LASzip* laszip;

    U32 user_data_after_header_size;
    U8* user_data_after_header;

    // load one after the other

    BOOL load_header(ByteStreamIn* stream);
    BOOL load_vlrs(ByteStreamIn* stream);
    BOOL load_evlrs(ByteStreamIn* stream);

    // convenience function

    const LASvlr* get_vlr(const char* user_id, U16 record_id) const;

    // housekeeping

    void clean();
    LASheader();
    ~LASheader();
};
```