MakeLabels

A PDS4 Label Generator Tool

PDS Geosciences Node

Version 6.2

5-01-2020

Created by Dan Scholes

scholes@wunder.wustl.edu

Send question and comments

to Dan Scholes.

Table of Contents

Summary

This application is designed to merge a PDS4 label template with data records from a Microsoft Excel spreadsheet to produce a set of PDS4 labels. It replaces placeholders in the label template with values from the spreadsheet, creating one label for each row in the spreadsheet. In fact, the template does not need to be a PDS4 label template; it can be any text file containing placeholders.

Users will most often use one Excel worksheet as the data source, but the program can handle two worksheets linked by a common keyword. The two worksheets can be in the same Excel file or separate files. This capability is helpful in scenarios of one worksheet containing the primary information, with a secondary worksheet of information referenced by one or more rows of the first worksheet. For example, when dealing with spectral data, the primary worksheet might contain information about the spectrum, whereas the secondary worksheet contains information about the target specimen. Multiple spectrum entries from the primary worksheet may be associated with each specimen entry in the secondary worksheet. In database terms, this is referred to as a “one-to-many” relationship; in this case, one specimen may be associated with many spectra.

A graphical user interface (GUI) version of the program and a command line version are provided. Most users will use the GUI version for rapid or infrequent processing. The command line version is more useful when setting up automated processes. In the command line version, settings can be submitted from the command line or configured in the program's xml configuration file.

System Requirements

- Microsoft Windows 7 or greater desktop OS or
- Microsoft Windows 2008 or greater server OS
- Microsoft Excel 2013 or greater .xlsx (has not been tested with older versions)
- Microsoft .NET Framework 4.5.2 (64-bit) (included in this package)

Installation

To install, simply unzip the MakeLabels.zip file to a directory on your machine or server.

Directory Structure

```
MakeLabels
    Examples
        BasicOneSheet
        OneSheetSpectrum
        OneSheetTwoWorksheets
        TwoSheetSpectrum
    Program
        CommandLine
        Gui
        RequiredFiles
```

The base directory contains a readme file, which focuses on the usage of the command line application. The directory also contains this pdf document outlining instructions for both applications.

The “Examples” directory contains example templates, Excel files, and output labels. Instructions and the commands to generate the same output labels are provided below in the “Command Line Examples” section.

The "Program" directory contains the command line application, GUI, and required files. The "Gui" directory contains the graphical user interface (GUI) version of the program, which most users will prefer. The command line version can be more helpful when setting up a data processing pipeline.

The "RequiredFiles" directory contains the Microsoft .NET Framework 4.5.2 installation file, which is required for both versions to function. Windows 10 includes .NET Framework 4.5.2 preinstalled. Other Windows operating systems may require the .NET Framework 4.5.2 to be installed. The operating system's list of installed programs can be checked to confirm the installation of the .NET Framework 4.5.2. Otherwise, the label generator program will report an error indicating that the framework is missing, if it is not installed. If the .NET Framework needs to be installed, simply double click the provided installation file and follow the instructions. More information about the .NET Framework 4.5.2 can be found here: https://www.microsoft.com/en-us/download/details.aspx?id=42642

Building the Worksheet(s)

Spreadsheet Structure

As previously mentioned, this application can be executed with one or two worksheets: a primary sheet and a secondary sheet. The worksheets may be in the same file or separate files.

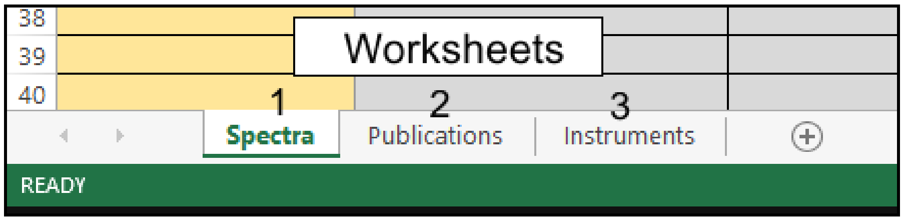
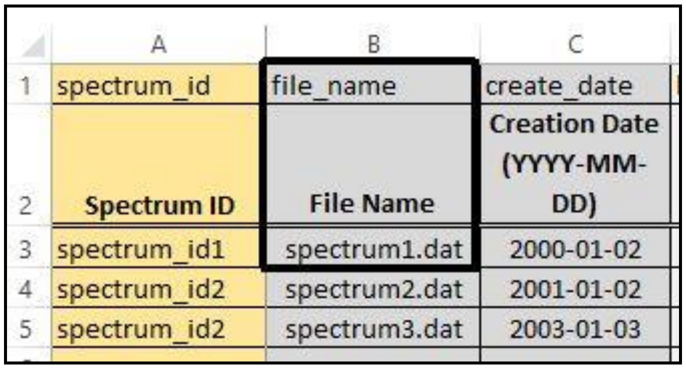


Figure 1: Example worksheets in one spreadsheet.

Each worksheet must follow the same basic structure. The first and second rows contain the column names in two forms. The first row is used by the application to find the template placeholders for the corresponding values of this column. These names must match between the template and worksheet. That being said, not every column must be used in the template. The second row is for your use and can follow any format that helps you know what data should go in each column. The application expects data to begin on the third row.



	A	B	C
1	spectrum_id	file_name	create_date
2	Spectrum ID	File Name	Creation Date (YYYY-MM-DD)
3	spectrum_id1	spectrum1.dat	2000-01-02
4	spectrum_id2	spectrum2.dat	2001-01-02
5	spectrum_id2	spectrum3.dat	2003-01-03

Figure 2: Example worksheet structure. Row 1: template placeholder name. Row 2: more detailed column name. Row 3: data begins.

In the primary worksheet, each row corresponds to one label that will be created. In the secondary worksheet, the information in a single row may be used for multiple labels.

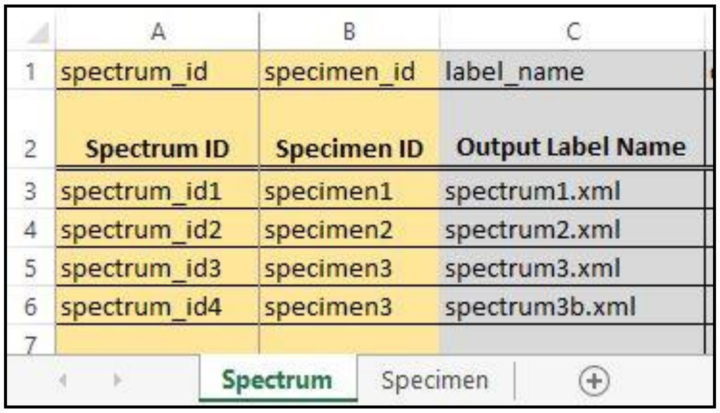
A column of the primary worksheet must provide the file names for the output labels. The GUI and command line versions of the application default to a column named “file_name”. An example is displayed in Figure 2. If a file extension is included in the field value, it will be replaced with “.xml”, but no file extension is required in the column. The default file name column can be changed through the GUI or the application’s configuration file. The output file’s extension of “.xml” can be modified in the application’s configuration file, as well. More about the configuration file will be discussed later in this document.

Requirements

- The first cell in the spreadsheet (row 1, column 1) must contain a column name, as it appears in a placeholder in the label template.
- Column names must be entirely lower case and be on row 1 of the worksheet.
- Data begins on row 3 of the spreadsheet.
- A “file_name” or equivalent column is included in the primary spreadsheet.

Using Two Sheets

When using both a primary and a secondary worksheet, there a couple of other guidelines to follow.

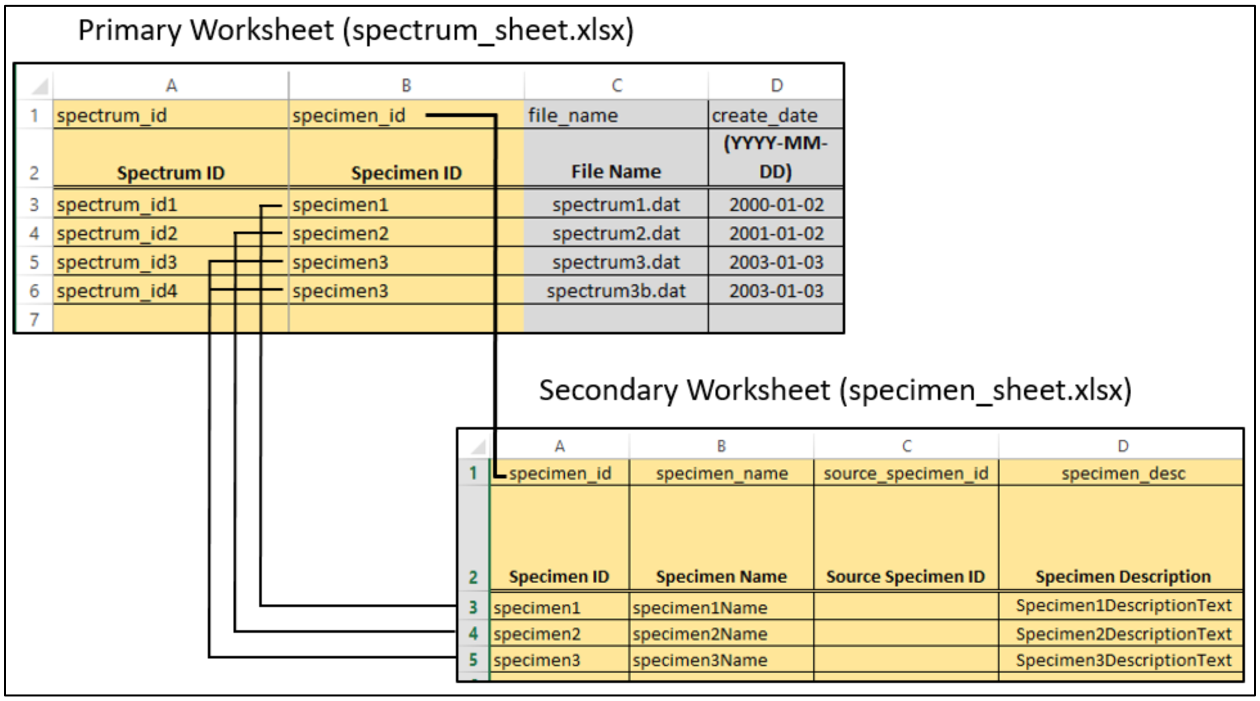


	A	B	C
1	spectrum_id	specimen_id	label_name
2	**Spectrum ID**	**Specimen ID**	**Output Label Name**
3	spectrum_id1	specimen1	spectrum1.xml
4	spectrum_id2	specimen2	spectrum2.xml
5	spectrum_id3	specimen3	spectrum3.xml
6	spectrum_id4	specimen3	spectrum3b.xml
7			

Figure 3: Two worksheets in one Excel spreadsheet. “Spectrum” is the primary sheet. “Specimen” is the secondary sheet.

First, the worksheet designated as the primary sheet should be the worksheet with the output label name column, such that each row will correspond to one label.

Second, the two worksheets must have a “matching field” that links them together. The following figure displays an example of this one-to-many relationship.



Primary Worksheet (spectrum_sheet.xlsx)

	A	B	C	D
1	spectrum_id	specimen_id	file_name	create_date
2	Spectrum ID	Specimen ID	File Name	(YYYY-MM-DD)
3	spectrum_id1	specimen1	spectrum1.dat	2000-01-02
4	spectrum_id2	specimen2	spectrum2.dat	2001-01-02
5	spectrum_id3	specimen3	spectrum3.dat	2003-01-03
6	spectrum_id4	specimen3	spectrum3b.dat	2003-01-03
7				

Secondary Worksheet (specimen_sheet.xlsx)

	A	B	C	D
1	specimen_id	specimen_name	source_specimen_id	specimen_desc
2	Specimen ID	Specimen Name	Source Specimen ID	Specimen Description
3	specimen1	specimen1Name		Specimen1DescriptionText
4	specimen2	specimen2Name		Specimen2DescriptionText
5	specimen3	specimen3Name		Specimen3DescriptionText

Figure 4: Secondary worksheet. In this case, the matching field is the “specimen_id”.

Building the Template

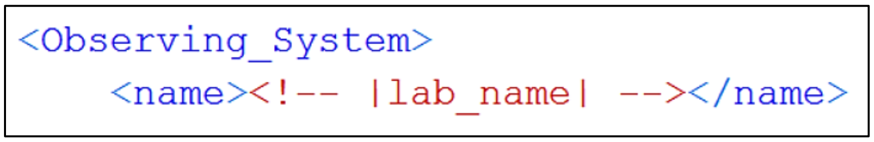
Template Structure

This guide is not intended to provide a blueprint for building a PDS4 label, but rather the label template that the tool will use to complete your labels with the proper PDS4 information. Using a previously created sample label, you will be able to prepare a template for use with the tool.

You will use a “template tag” to indicate placeholders in the template that should be populated from a worksheet. This is very similar to a mail merge template. The “template tag” will take a couple of different forms, depending on its purpose.

Possible Template Tags

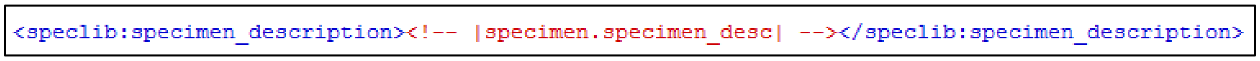
Populate a value from primary worksheet. This is the most common tag type to place in a template. The column name from the worksheet’s first row is wrapped with the opening and closing tags. For the column name “lab_name”, the syntax would be “`<!-- |lab_name| -->`”. The exact characters and spaces are required.



```
<Observing_System>
    <name><!-- |lab_name| --></name>
```

Figure 5: A value from the spreadsheet column named “lab_name” in the primary worksheet will be used to replace the text shown in red.

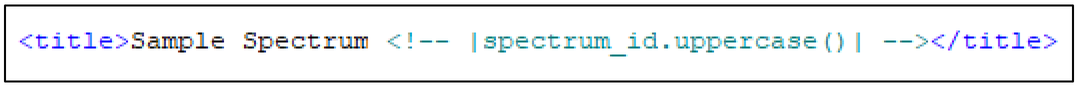
Populate a value from the secondary worksheet. To populate the template with a value from the secondary worksheet a similar tag must be applied around the column name, with the addition of a template field qualifier. The template field qualifier is a modifier to indicate columns in the secondary worksheet. The template field qualifier is set from the GUI or in the application’s configuration file. This will be discussed later. For the secondary worksheet column “specimen_desc”, the syntax would be “`<!-- |specimen.specimen_desc| -->`”.



```
<speclib:specimen_description><!-- |specimen.specimen_desc| --></speclib:specimen_description>
```

Figure 6: A value from the column named “specimen_desc” in the secondary worksheet “specimen” will be used to replace the text shown in red.

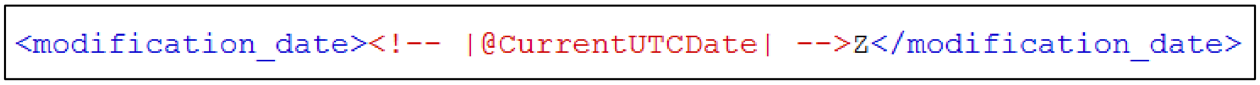
Output upper or lower case value. To force an output column value to be upper or lower case in the produced label, the following syntax can be added to the two previous examples. For lower case, use “`<!-- |column_name.lowercase()| -->`”. To output the field in upper case, use “`<!-- |column_name.uppercase()| -->`”. This tag can be added to secondary worksheet values, such as “`<!-- |specimen.instrument_abreviation.lowercase()| -->`”. The upper and lower case tags are **case sensitive.**



```
<title>Sample Spectrum <!-- |spectrum_id.uppercase()| --></title>
```

Figure 7: An upper case tag is used to force a value to output in upper case.

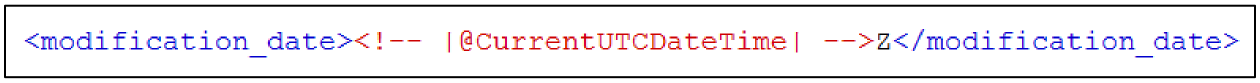
Populate the current UTC date in the form of YYYY-MM-DD. To populate the template with the current coordinated universal time (UTC) date value (at the time of producing the label), use the following syntax. UTC dates and times are required for some PDS4 date/time attributes.
"`<!-- |@CurrentUTCDate| -->`"



```
<modification_date><!-- |@CurrentUTCDate| -->Z</modification_date>
```

Figure 8: Example of current UTC date.

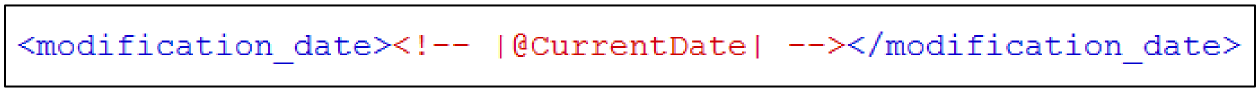
Populate the current UTC date and time in the form of YYYY-MM-DDThh:mm:ss To populate the template with the current coordinated universal time (UTC) date and time value (at the time of producing the label) to the second, use the following syntax.
"`<!-- |@CurrentUTCDateTime| -->`"



```
<modification_date><!-- |@CurrentUTCDateTime| -->Z</modification_date>
```

Figure 9: Example of current UTC date and time.

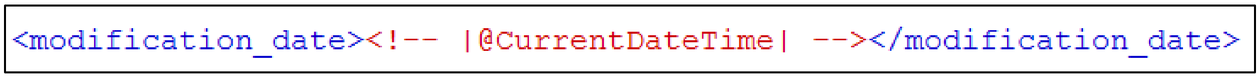
Populate the current date in the form of YYYY-MM-DD. To populate the template with the current local system date, use the following syntax.
"`<!-- |@CurrentDate| -->`"



```
<modification_date><!-- |@CurrentDate| --></modification_date>
```

Figure 10: Example of current local date.

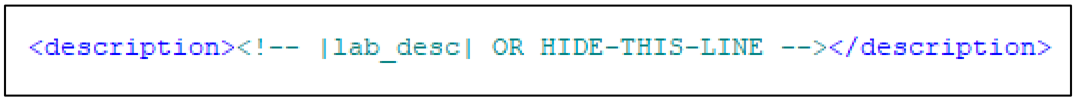
Populate the current date and time in the form of YYYY-MM-DDThh:mm:ss To populate the template with the current local date and time value to the second, use the following syntax.
"`<!-- |@CurrentDateTime| -->`"



```
<modification_date><!-- |@CurrentDateTime| --></modification_date>
```

Figure 11: Example of current local date and time.

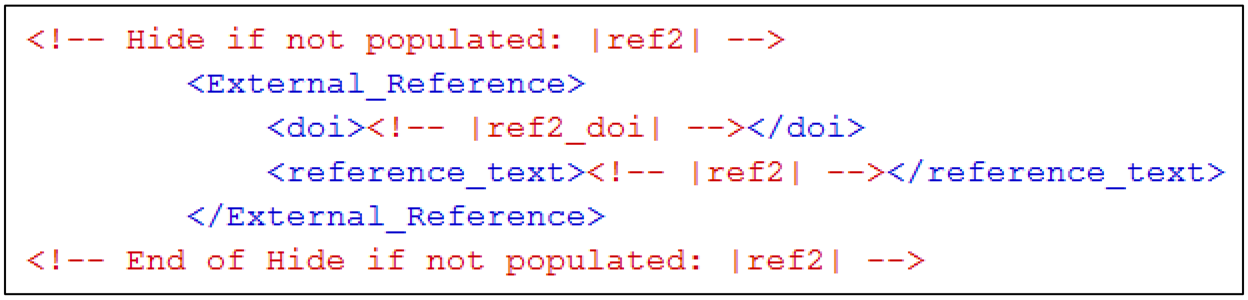
Include a template line if the referenced column value is populated. If your template has a line that should only be included if a particular spreadsheet column is populated or exists, use the syntax "`<!-- |column_name| OR HIDE-THIS-LINE -->`". The "HIDE-THIS-LINE" is case sensitive. If the column does not contain a value, the entire line will be omitted from the output label.



```
<description><!-- |lab_desc| OR HIDE-THIS-LINE --></description>
```

Figure 12: Example of applying the option to hide a line if a specified field is not populated.

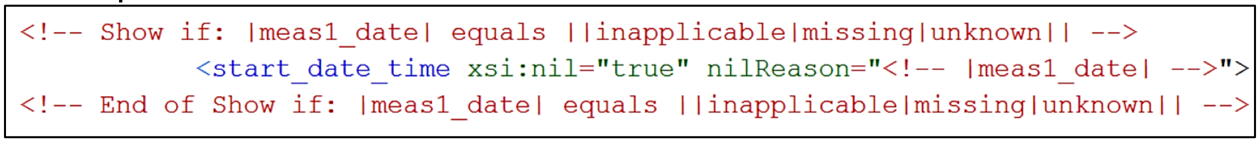
Create a label section only if a column value is populated. If your template has a section (such as an External_Reference) that will not always be filled, use the following tags to surround the section. These tags essentially act as an "if" statement. Make sure there are no trailing spaces after the closing tags of "|column_name| -->", otherwise blanks lines may appear in the output labels.



```
<!-- Hide if not populated: |ref2| -->
        <External_Reference>
            <doi><!-- |ref2_doi| --></doi>
            <reference_text><!-- |ref2| --></reference_text>
        </External_Reference>
<!-- End of Hide if not populated: |ref2| -->
```

Figure 13: Example of second external reference column that may or may not be filled in a worksheet.

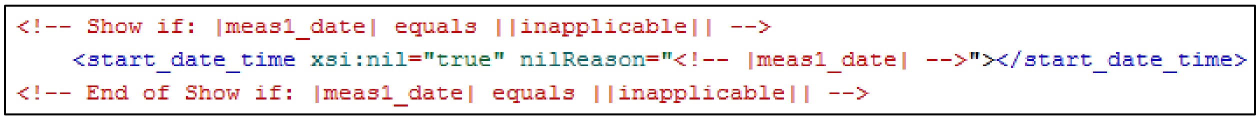
Create a label section only if a column value equals a specified value. If your template has a section that should be included only when specific values occur, use the following tags to surround the section. These tags act as an "if" statement. Make sure there are no trailing spaces after the closing tags of "|matching value| -->", otherwise blanks lines may appear in the output labels.



```
<!-- Show if: |meas1_date| equals ||inapplicable|missing|unknown|| -->
             <start_date_time xsi:nil="true" nilReason="<!-- |meas1_date| -->">
<!-- End of Show if: |meas1_date| equals ||inapplicable|missing|unknown|| -->
```

Figure 14: Example of a show if column equals a specified value.

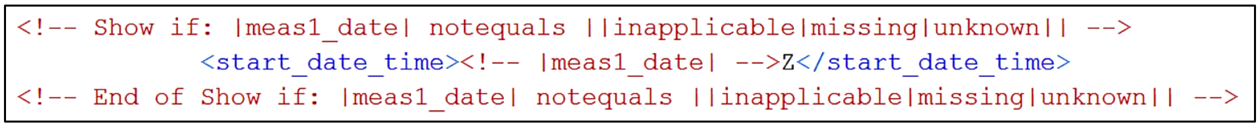
Multiple matching values may be included inside the double vertical bars. Each value is delimited with a single vertical bar. If only one value is provided, it is enclosed in the double vertical bars.



```
<!-- Show if: |meas1_date| equals ||inapplicable|| -->
    <start_date_time xsi:nil="true" nilReason="<!-- |meas1_date| -->"></start_date_time>
<!-- End of Show if: |meas1_date| equals ||inapplicable|| -->
```

Figure 15: Example of providing a single optional value.

Create a label section only if a column value does not equal a specified value. This syntax is the same as the previous section with the exception of the conditional expression of “notequals”. Multiple matching values or a single value may be provided.



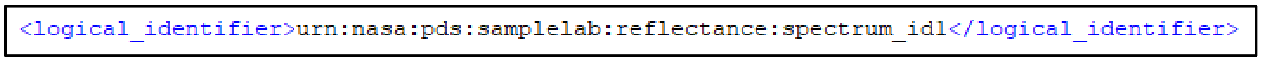
```
<!-- Show if: |meas1_date| notequals ||inapplicable|missing|unknown|| -->
            <start_date_time><!-- |meas1_date| -->Z</start_date_time>
<!-- End of Show if: |meas1_date| notequals ||inapplicable|missing|unknown|| -->
```

Figure 16: Example of a show if column does not equal a provided value.

Miscellaneous

PDS4 Logical Identifier Requirements

- Every PDS4 product must have a unique logical identifier (LID) that follows a specific format. (See your PDS contact or the PDS Data Provider’s Handbook, for more information.)
- A product’s LID must be completely lowercase. The worksheet column that will fill the LID in the template must therefore be completely lowercase as well.
- The full LID may be no longer than 255 characters.



```
<logical_identifier>urn:nasa:pds:samplelab:reflectance:spectrum_id1</logical_identifier>
```

Figure 17: Example LID from a sample label.

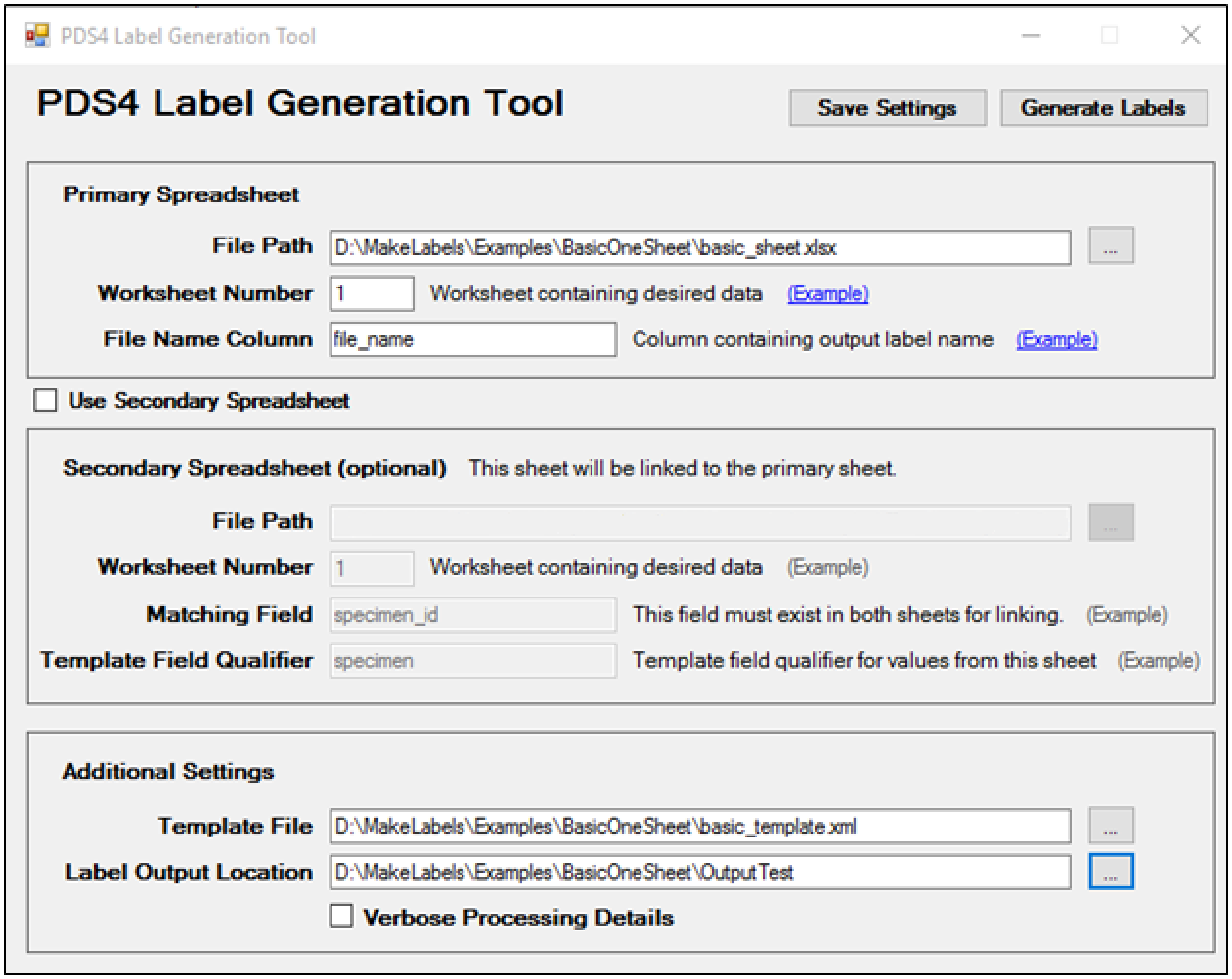
PDS 4 Date Requirements

- The only acceptable format for “start_date_time” and “stop_date_time” is yyyy-mm-ddThh:mm:ss.fffZ where
    - yyyy = year
    - mm = month (01 through 12)
    - dd = day (01 through 31)

  - T is just the letter T, always capitalized
  - hh = hour
  - mm = minute
  - ss = second
  - fff = fraction of a second
  - Z is the letter Z, which means UTC time
- All data observations must be UTC times, so Z is required for those attributes.
- All numeric fields must include leading zeroes as needed.
- You can truncate times from the right, if you only know the hour and minute, for example.
- You can leave off the T and the time entirely if you only know the date.
- You can also truncate dates from the right, if you only know the year, for example.

GUI Interface

To open the GUI, double click on the file “guiMakeLabels.exe” in the “Program\Gui” directory. The interface below will be launched.



Primary Spreadsheet

File Path	D:\MakeLabels\Examples\BasicOneSheet\basic_sheet.xlsx	
Worksheet Number	1	Worksheet containing desired data (Example)
File Name Column	file_name	Column containing output label name (Example)

☐ Use Secondary Spreadsheet

Secondary Spreadsheet (optional) This sheet will be linked to the primary sheet.

File Path		
Worksheet Number	1	Worksheet containing desired data (Example)
Matching Field	specimen_id	This field must exist in both sheets for linking. (Example)
Template Field Qualifier	specimen	Template field qualifier for values from this sheet (Example)

Additional Settings

Template File	D:\MakeLabels\Examples\BasicOneSheet\basic_template.xml
Label Output Location	D:\MakeLabels\Examples\BasicOneSheet\OutputTest

☐ Verbose Processing Details

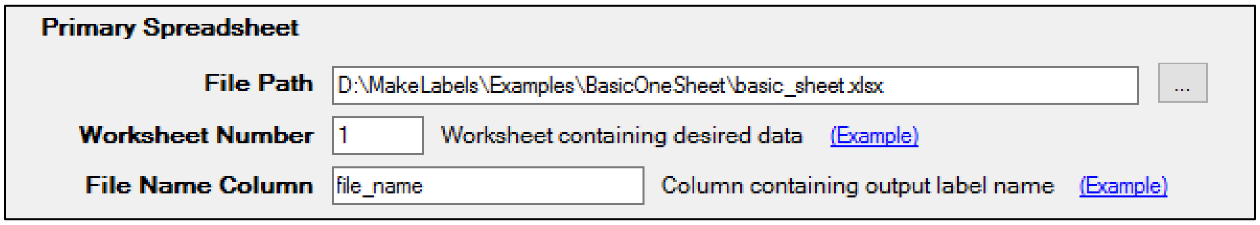
Figure 18: MakeLabels GUI interface.

The same values that can be submitted from the command line are available through this interface.

Setting up the Primary Sheet

First, you must enter the path to the primary spreadsheet and set the worksheet to be processed.

Additionally, the column containing the filenames to be used for the output labels must be set. The default column name is “file_name”. If a file extension is included in the field value, it will be replaced with .xml, but no extension is required.


Primary Spreadsheet

File Path	D:\MakeLabels\Examples\BasicOneSheet\basic_sheet.xlsx	
Worksheet Number	1	Worksheet containing desired data (Example)
File Name Column	file_name	Column containing output label name (Example)

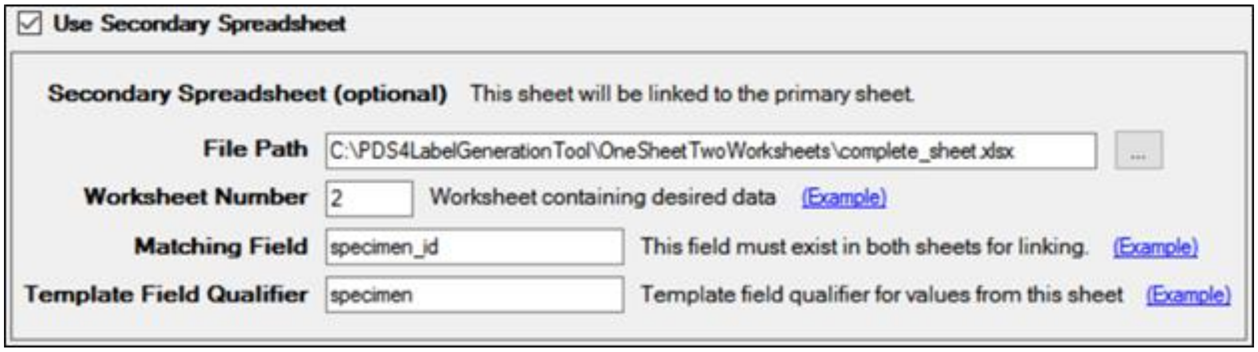
Figure 19: Section for primary sheet setup.

Adding a Secondary Sheet

If a secondary spreadsheet is used, the corresponding section must be populated. First, check the “Use Secondary Spreadsheet” option to enable this section of the form.

The corresponding file path and worksheet number are required. Additionally, the matching field, which was previously discussed, must be set. This is the column that must exist in both worksheets, so the rows can be linked.

The template field qualifier is also set at this location. It is the modifier that will precede the column name in the template tag to indicate columns in the secondary worksheet. For example, the value “specimen” will be used for the template tag “`<!-- |specimen.specimen_desc| -->`”.


☑ Use Secondary Spreadsheet

Secondary Spreadsheet (optional) This sheet will be linked to the primary sheet.

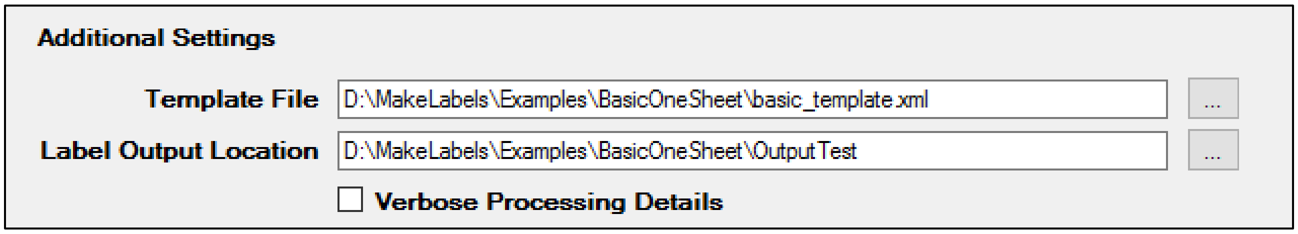
File Path	C:\PDS4LabelGenerationTool\OneSheetTwoWorksheets\complete_sheet.xlsx	
Worksheet Number	2	Worksheet containing desired data (Example)
Matching Field	specimen_id	This field must exist in both sheets for linking. (Example)
Template Field Qualifier	specimen	Template field qualifier for values from this sheet (Example)

Figure 20: Section for secondary sheet setup.

Additional Settings

The full path to the template file must be specified. The desired output directory for the created labels must be set, as well.

An option for verbose processing details is available. If selected, when the "Generate Labels" button is clicked, a status will be displayed for each record that is processed. Otherwise, only a processing summary will be displayed.



Template File: D:\MakeLabels\Examples\BasicOneSheet\basic_template.xml

Label Output Location: D:\MakeLabels\Examples\BasicOneSheet\OutputTest

☐ Verbose Processing Details

Figure 21: Section for additional settings

Generating Labels through the GUI

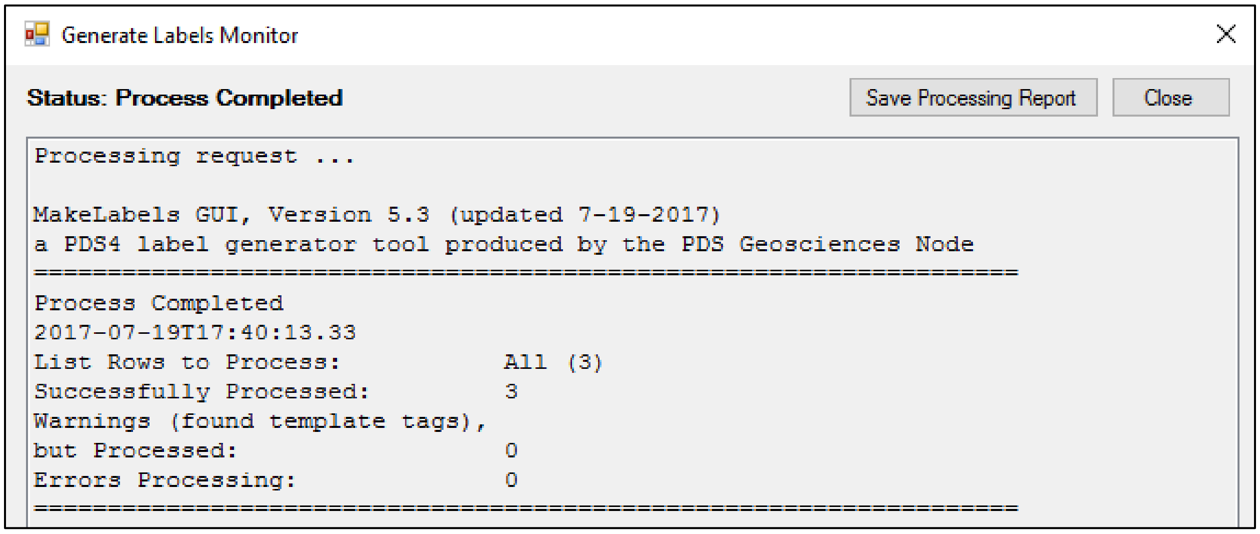
After values have been set, you can save the values for later by clicking the "Save Settings" button. Your settings will be available the next time the application is opened.

To produce the labels, click the "Generate Labels" button. Your settings will be automatically saved.

You will receive a descriptive error message if the program detects any invalid entries in the form when you try to save the settings or generate labels.

After the process has completed, the status of the process is displayed. The number of successfully processed worksheet rows are listed, in addition to errors and warnings. More details will be produced by selecting the verbose option.

The processing report can be saved as a text file with the "Save Processing Report" button.



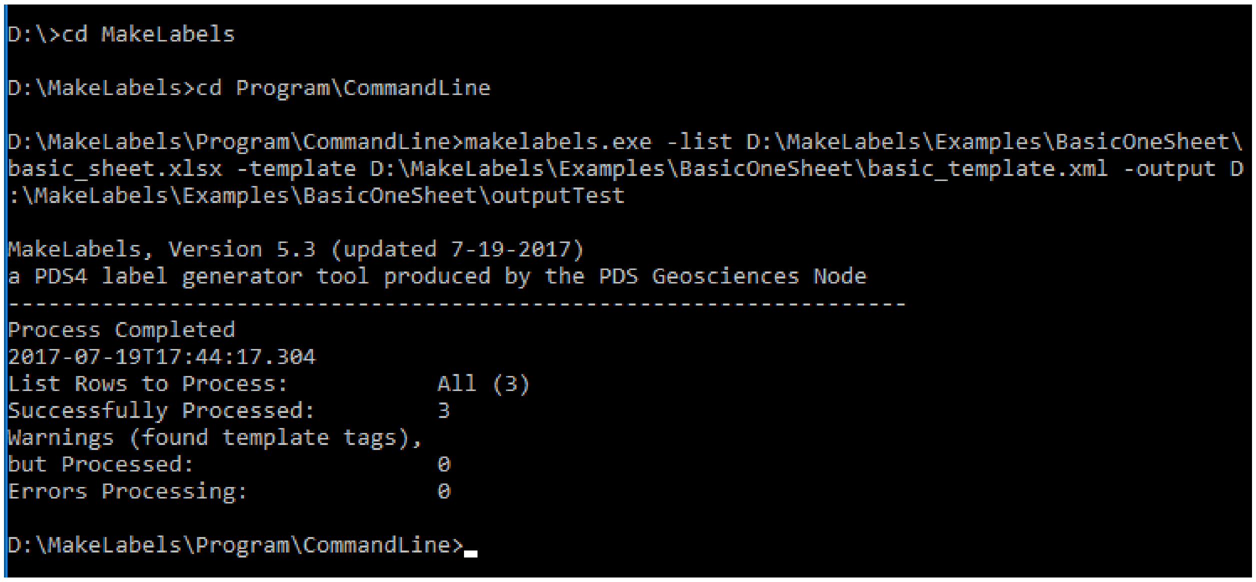
```
Processing request ...

MakeLabels GUI, Version 5.3 (updated 7-19-2017)
a PDS4 label generator tool produced by the PDS Geosciences Node
===================================================================
Process Completed
2017-07-19T17:40:13.33
List Rows to Process:           All (3)
Successfully Processed:         3
Warnings (found template tags),
but Processed:                  0
Errors Processing:              0
===================================================================
```

Figure 22: Processing status screen

The labels that were produced should be in the output directory you set on the previous form.

Command Line Program

To run the application from the command line, you must make the “CommandLine” folder your current directory. From there, the program is run by calling MakeLabels.exe, followed by keyword value pairs. The command should be placed on one line, although the command window may wrap the text.



```
D:\>cd MakeLabels

D:\MakeLabels>cd Program\CommandLine

D:\MakeLabels\Program\CommandLine>makelabels.exe -list D:\MakeLabels\Examples\BasicOneSheet\
basic_sheet.xlsx -template D:\MakeLabels\Examples\BasicOneSheet\basic_template.xml -output D
:\MakeLabels\Examples\BasicOneSheet\outputTest

MakeLabels, Version 5.3 (updated 7-19-2017)
a PDS4 label generator tool produced by the PDS Geosciences Node
-------------------------------------------------------------------
Process Completed
2017-07-19T17:44:17.304
List Rows to Process:          All (3)
Successfully Processed:        3
Warnings (found template tags),
but Processed:                 0
Errors Processing:             0

D:\MakeLabels\Program\CommandLine>_
```

Figure 23: Command line processing screen capture

Command Line Parameters

Each parameter is a combination of **-keyword value**. All of these parameters can be declared on the command line or changed in the program’s configuration file (MakeLabels.exe.config). References to "**full path**" refer to an absolute file path including the drive letter or a Windows UNC (Universal Naming Convention) path. Examples:
c:\MakeLabels\example\file.xyz or
\\serverName\MakeLabels\example\file.xyz

-list
The full path to the primary spreadsheet (.xlsx file).
example: -list c:\tempLocation\primaryFile.xlsx

-worksheet
The primary spreadsheet's worksheet to be processed. The application defaults to 1 (that is, the first worksheet in the spreadsheet file) if no value is provided at the command line or in the configuration file.
example: -worksheet 1

The following three parameters are only for two-sheet configurations.

-secondaryList
The full path to the secondary spreadsheet (.xlsx file). This must be included, even if it is the same as the primary spreadsheet.
example: -secondaryList c:\tempLocation\secondaryFile.xlsx

-secondaryWorksheet
The secondary spreadsheet's worksheet to be processed. The application defaults to 1 if no value is provided at the command line or in the configuration file.
example: -secondaryWorksheet 1

-secondaryMatchingField
The field used to link the two sheets. This column needs to be in both spreadsheet worksheets.
example -secondaryMatchingField specimen_id

-template
The full path to the template file to be used when executing the program. This value can also be set in the program's configuration file under the <defaultXMLTemplateFile> tag.
example: -template c:\tempLocation\templateFile.xml

-output
The directory to output the newly created labels. This directory must exist or an error message will be displayed.
example: -output c:\tempLocation\files\

Important additional note: The command line version of the MakeLabels tool expects the primary spreadsheet's worksheet to contain a column named "file_name". This column is used to indicate each output label's file name. The ".xml" file extension is attached to the file name or any provided extension is replaced with ".xml". The column name for this feature can be changed in the program's configuration file. The configuration key is "primaryFileNameColumn". The file extension for the output files (defaulting to ".xml") can be changed in the configuration file, as well. Its key is "defaultOutputFileExtension". The next section includes more information about the configuration file.

Updating the Configuration File

As previously mentioned, the command line parameters can also be set and retrieved from the program's configuration file. The file is named MakeLabels.exe.config. This file must be located in the same directory as the command line program (MakeLabels.exe). The program will first attempt to use any value submitted from the command line. If the parameters are not found, it will look to the configuration file for the information. Parameters submitted from the command line always take precedence over those found in the configuration file. We have included an additional example configuration file in this directory. If you wish to make changes to this file, we recommend using a text editor program.

The configuration file of the GUI version of MakeLabels can be modified, but most of its values are overwritten with user preferences each time the GUI produces labels. The file is named guiMakeLabels.exe.config. It is located in the same directory as the GUI interface (guiMakeLabels.exe).

Command Line Examples

For each of the following examples the executable program (MakeLabels.exe) and following commands should be placed on one line. You will most likely need to change the basic path provided to the correct path for your environment. The examples output sample labels to "OutputTest" directories, so you can compare your labels with the example labels provided in the "Output" directories.

BasicOneSheet

This basic sample contains an abbreviated template, abbreviated spreadsheet, and defaults to worksheet 1.

```
MakeLabels.exe
-list D:\MakeLabels\Examples\BasicOneSheet\basic_sheet.xlsx
```

```
-template D:\MakeLabels\Examples\BasicOneSheet\basic_template.xml
-output D:\MakeLabels\Examples\BasicOneSheet\outputTest
```

OneSheetSpectrum
This sample uses a full template and one spreadsheet.

```
MakeLabels.exe
-list D:\MakeLabels\Examples\OneSheetSpectrum\basic_sheet.xlsx
-worksheet 1
-template
D:\MakeLabels\Examples\OneSheetSpectrum\one_sheet_spectrum_templ
ate.xml
-output D:\MakeLabels\Examples\OneSheetSpectrum\outputTest
```

OneSheetTwoWorksheets
This sample uses a full template and one spreadsheet with two worksheets of data to be linked.

```
MakeLabels.exe
-list
D:\MakeLabels\Examples\OneSheetTwoWorksheets\complete_sheet.xlsx
-worksheet 1
-secondaryList
D:\MakeLabels\Examples\OneSheetTwoWorksheets\complete_sheet.xlsx
-secondaryWorksheet 2
-secondaryMatchingField specimen_id
-template
D:\MakeLabels\Examples\OneSheetTwoWorksheets\two_sheet_template.x
ml
-output D:\MakeLabels\Examples\OneSheetTwoWorksheets\outputTest
```

TwoSheetSpectrum
This sample uses a full template and two linked spreadsheets (spectrum + specimen).

```
MakeLabels.exe
-list D:\MakeLabels\Examples\TwoSheetSpectrum\spectrum_sheet.xlsx
-worksheet 1
-secondaryList
D:\MakeLabels\Examples\TwoSheetSpectrum\specimen_sheet.xlsx
```

```
-secondaryWorksheet 1
-secondaryMatchingField specimen_id
-template
D:\MakeLabels\Examples\TwoSheetSpectrum\two_sheet_template.xml
-output D:\MakeLabels\Examples\TwoSheetSpectrum\outputTest
```

Troubleshooting

It is recommended to save and close a referenced Excel spreadsheet before running the label generator. Closing the file is not absolutely required, but it is good practice.

Sometimes blank lines are output in labels from templates using the “Hide if” or “Show if” tags. This can occur if trailing blank spaces or tabs are left at the end of the lines of these tags.
Example: “<!-- Hide if not populated: |instr2_id| -->   ”

Questions can be directed to Dan Scholes (scholes@wunder.wustl.edu) at the PDS Geosciences Node.