**API for Running Inline Vespa in Philips PRIDE 2.0** June 8th, 2020

**Version – 0.1.0**

**General workflow:**

Vespa inline processing on the Philips platform consists of two primary modules.

* run\_inline\_vespa\_philips.py – all the scanner specific code.
* vespa\_inline\_engine.py – all the scanner independent, Vespa code.

The main call to process data is to run\_inline\_vespa\_philips.run() which requires that a ‘settings’ object be passed in. This is an instantiation of vespa\_inline\_engine.VespaInlineSettings(). It contains a variety of paths, filename, flags and output parameters to control what data is processed and what outputs are created. This includes, also, any error reporting directives if fitting fails.

A general workflow might include:

1. Create your own Python script to call run\_inline\_vespa\_philips.run() method.
2. Instantiate a vespa\_inline\_engine.VespaInlineSettings() object.
3. Fill in the settings values that point to the data, output, preset and debug directories, and other output options and parameter values.
   1. See class definition below for documentation on settings.
   2. At a minimum, you should turn on the DICOM output.
   3. If possible, set dcm\_dpi and err\_dpi to 200 for clarity, this creates a 2048x2048 image.
   4. Both dcm\_dpi and err\_dpi should match the png\_dpi setting.
   5. Remember, you can use the settings.base\_path string to get the absolute path to where the vespa\_inline\_engine.py module resides. This may help you set the other paths.
4. Call run\_inline\_vespa\_philips.run() method with required ‘settings’ object.
5. This returns a pydicom.FileDataset object. You can save this yourself, or use the settings.save\_dcm options to output this object to a specific location before it returns to you.

**VespaInlineEngine.Settings object Class Definition**

**class** VespaInlineSettings:

**def** **\_\_init\_\_**(self):

'''

This class embodies a list of input/output settings that control the

behaviour of the Vespa Inline Engine (VIE), and to some extent the scanner

dependent code module that calls the VIE. It has default values that

should work for a standard setup, but can be tailored by the user to

perform for a variety of other workflow locations.

'''

self.vespa\_version = <auto-set>

self.base\_path = <auto-set> # directory in which this module resides

self.import\_class = <auto-set> # set in the data-parsing step

self.dataformat = '' # user must provide – see documentation below

# Input/Output directory paths (absolute) where data can be found or results output

# - coding trick, self.base\_path is auto-set on instantiation to directory in which

# this module resides. This may also be where sub-directories reside for presets,

# output, debug, data, etc.

# - Ex. self.preset\_dir = os.path.join(self.base\_path, 'presets')

self.data\_dir = ''

self.preset\_dir = ''

self.output\_dir = ''

self.debug\_dir = ''

# Output flags - control output produced, True means output is created

# xml - full provenance of vespa-analysis fitting, can load into Vespa-Analysis GUI

# pdf – hires image of plot/table fitting results

# png – image of plots/table for fitting results – same as in DICOM output

# dcm – ‘screenshot’ image of plots/table for fitting results

# err – text output of filenames and traceback due to fitting error

self.save\_xml = False

self.save\_pdf = False

self.save\_png = False

self.save\_dcm = True

self.save\_err = True

# Flags for unique file naming

# - if True, append timestamp to any given filename just before the extension

# - timestamp includes: yymmdd.hhmmss.\_usec\_

# - Ex. dicom\_out\_20200603.163207.478473.dcm

self.xml\_fname\_unique = True

self.pdf\_fname\_unique = True

self.png\_fname\_unique = True

self.dcm\_fname\_unique = True

self.err\_fname\_unique = True

# Output filenames

# - should be absolute path/filename.extn

# - can use self.debug\_dir, self.output\_dir to create these

# - Ex. self.png\_fname = os.path.join(self.output\_dir,'output\_png.png')

self.xml\_fname = ''

self.pdf\_fname = ''

self.png\_fname = ''

self.dcm\_fname = ''

self.err\_fname = ''

# The following sections set parameters used to create the pdf, png and dcm images.

# Many correspond to keywords used by matplotlib (which is used to create the images)

# PDF results plot settings -------------------------------------------

self.pdf\_plotstyle = 'lcm\_multi' # see documentation below

self.pdf\_file\_label = 'default\_pdf' # part of the title on Figure

self.pdf\_inbuf = False # deprecated

self.pdf\_minppm = 0.5 # min/max plot range in ppm

self.pdf\_maxppm = 4.2

self.pdf\_apply\_phase = False # phase spectrum to improve plot clarity

self.pdf\_remove\_base = False # subtract baseline from spectral plot

self.pdf\_fontname = 'Courier New' # font used in Figure

self.pdf\_dpi = 300 # output file resolution

self.pdf\_pad\_inches = 0.5 # affects outer border on output file

# PNG and DICOM results file settings ------------------------------------

self.png\_plotstyle = 'lcm\_square' # see documentation below

self.png\_file\_label = 'default\_png' # part of the title on Figure

self.png\_inbuf = False # deprecated

self.png\_minppm = 0.5 # min/max plot range in ppm

self.png\_maxppm = 4.2

self.png\_apply\_phase = False # phase spectrum to improve plot clarity

self.png\_remove\_base = False # subtract baseline from spectral plot

self.png\_fontname = 'Courier New' # font used in Figure

self.png\_dpi = 200 # output file resolution

self.png\_pad\_inches = 0.5 # affects outer border on output file

# Note. The PNG settings also determine the DICOM output. The PNG Figure is defined

# to be 10.24 x 10.24 inches, so a png\_dpi = 100 would result in a 1024x1024 DICOM

# image resolution.

# Error reporting output settings ---------------------------------------

# - These settings also affect the size of a DICOM result image, just a debug one.

# - A err\_dpi = 100 would result in a 1024x1024 DICOM image.

# - If possible, the text is much clearer at err\_dpi = 200, but …

self.err\_dpi = 100

self.err\_pad\_inches = 0.5

# Generic flag for other debug actions. At the moment, setting to True will output

# the PNG buffer, that becomes the DICOM result, into a numpy save file using

# ndarray.tofile() method in same directory as self.png\_fname

self.debug = False

**Entries for** self.dataformat

Allowable settings for this parameter: 'philips\_press28\_dicom' and 'philips\_slaser30\_cmrr\_spar'

**Entries for** self.pdf\_plotstyle

Allowable settings for this parameter: 'lcm' and 'lcm\_multi'

**Entries for** self.png\_plotstyle

Allowable settings for this parameter: 'lcm', 'lcm\_square', 'lcm\_multi' and 'brp\_generic'

**Example:**

**import** vespa\_inline\_engine **as** vie

**def** my\_script():

settings = vie.VespaInlineSettings()

settings.dataformat = 'import\_philips\_dicom'

settings.save\_xml = True

settings.save\_pdf = True

settings.save\_png = True

settings.save\_dcm = True

settings.save\_err = True

settings.xml\_fname\_unique = True

settings.pdf\_fname\_unique = True

settings.png\_fname\_unique = True

settings.dcm\_fname\_unique = True

settings.err\_fname\_unique = True

settings.xml\_fname = os.path.join(settings.debug\_dir, "debug\_xml\_last\_run.xml")

settings.pdf\_fname = os.path.join(settings.debug\_dir, "debug\_pdf\_philips.pdf")

settings.png\_fname = os.path.join(settings.output\_dir,"results\_philips.png")

settings.dcm\_fname = os.path.join(settings.output\_dir,"output\_dicom\_last\_run.dcm")

settings.err\_fname = os.path.join(settings.debug\_dir, "debug\_vespa\_viff.xml")

settings.pdf\_plotstyle = 'lcm\_multi'

settings.pdf\_file\_label = 'Analysis- Philips PRIDE Inline'

settings.pdf\_minppm = 0.5

settings.pdf\_maxppm = 4.2

settings.pdf\_apply\_phase = False

settings.pdf\_remove\_base = False

settings.pdf\_fontname = 'Courier New'

settings.pdf\_dpi = 300

settings.pdf\_pad\_inches = 0.5

settings.png\_plotstyle = 'lcm\_square'

settings.png\_file\_label = 'Analysis- Philips PRIDE Inline'

settings.png\_minppm = 0.5

settings.png\_maxppm = 4.2

settings.png\_apply\_phase = False

settings.png\_remove\_base = False

settings.png\_fontname = 'Courier New'

settings.png\_dpi = 200

settings.png\_pad\_inches = 0.5

settings.err\_dpi = 100

settings.err\_pad\_inches = 0.5

vie.run(settings)