

```

import qupath.lib.gui.tools.MeasurementExporter
import qupath.lib.objects.PathAnnotationObject
import qupath.lib.roi.ShapeSimplifier

// Create shape simplifier
def simplifier = new ShapeSimplifier()

// Separate each measurement value in the output file with a tab ("\t")
def separator = ";"

// Choose the columns that will be included in the export
// Note: if 'columnsToInclude' is empty, all columns will be included
def columnsToInclude = new String[]{"Image","Name","ECClass: Collagen %","ECClass: Collagen area
Åµm^2","ECClass: Elastin %","ECClass: Elastin area Åµm^2","Area Åµm^2"}

// Choose the type of objects that the export will process
// Other possibilities include:
// 1. PathAnnotationObject
// 2. PathDetectionObject
// 3. PathRootObject
// Note: import statements should then be modified accordingly
def exportType = PathAnnotationObject.class

def imagesToExport = [getProjectEntry()]
def imageData = getCurrentImageData()
def hierarchy = imageData.getHierarchy()
def annotations = hierarchy.getAnnotationObjects()
float elevation = 10.0

// Choose your *full* output path
def name = GeneralTools.getNameWithoutExtension(imageData.getServer().getMetadata().getName())
def outputPath = "/nb_projects/qupath_Projects/zoneDetection/measurements/" + name + "_ELS-
CLN_measurements.csv"
def outputFile = new File(outputPath)

print "Image: " + name

hierarchy.getSelectionModel().clearSelection()
for (annotation in annotations) {
    annotation.setROI(ShapeSimplifier.simplifyShape(annotation.getROI(), elevation))
    hierarchy.getSelectionModel().setSelectedObject(annotation)
    runPlugin('qupath.lib.plugins.objects.RefineAnnotationsPlugin', '{"minFragmentSizeMicrons": 120.0,
"maxHoleSizeMicrons": 120.0}')
    print annotation.getPathClass()
    if (["Zone 1", "Zone 2", "Thrombus"].contains(annotation.getPathClass().name) ) {
        addPixelClassifierMeasurements("ECClass", "ECClass")
    }
}
// Save the detections before exporting
getProjectEntry().saveImageData(imageData)

// Create the measurementExporter and start the export
def exporter = new MeasurementExporter()

```

```
.imageList(imagesToExport)      // Images from which measurements will be exported
.separator(separator)           // Character that separates values
.includeOnlyColumns(columnsToInclude) // Columns are case-sensitive
.exportType(exportType)         // Type of objects to export
.exportMeasurements(outputFile) // Start the export process
```

```
print "Done!"
```