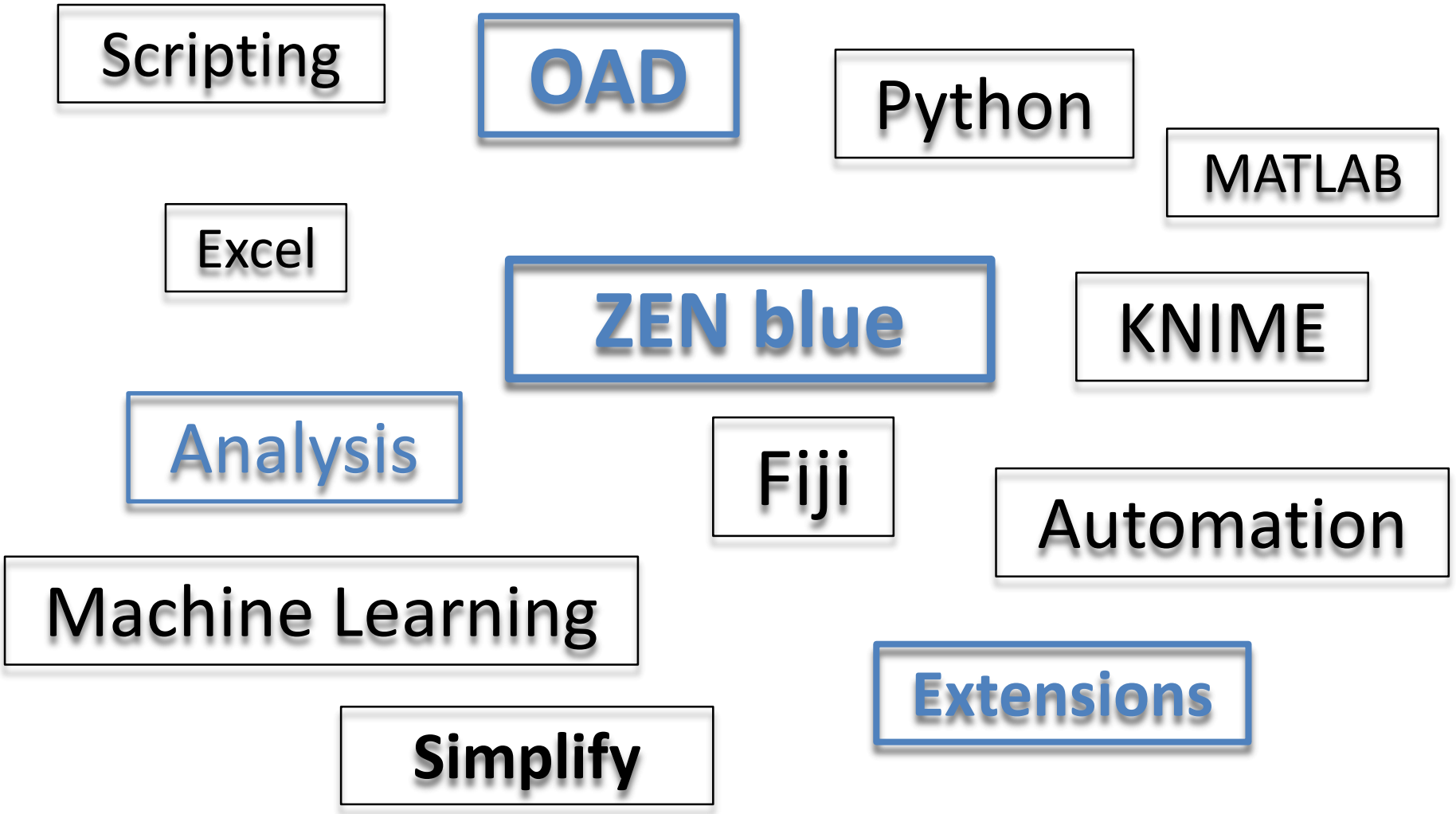
Experiment Feedback

Dr. Marion Lang
Product Management
2018-10-05

ZEN is only part of the workflow

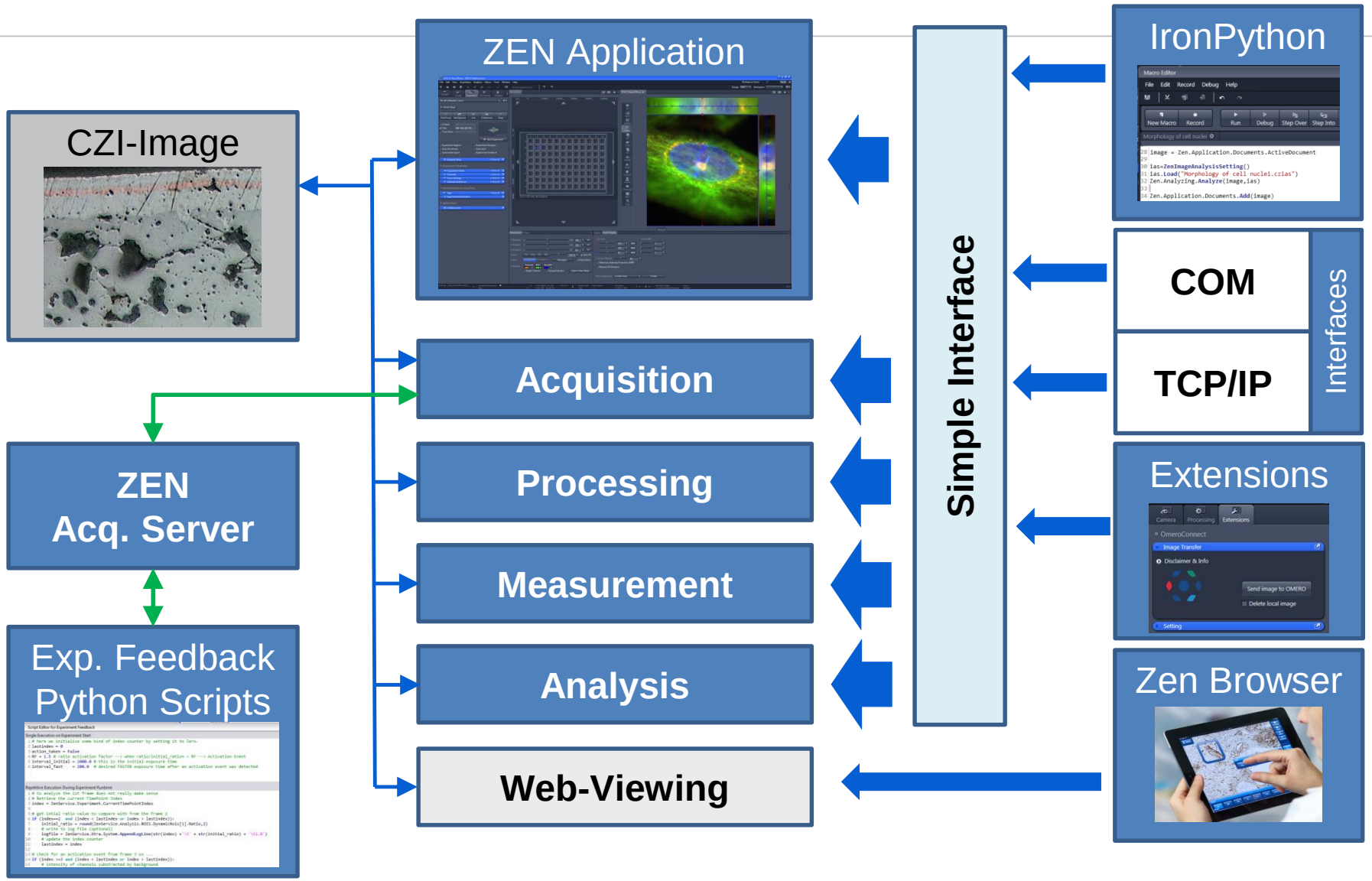


„Feedback Experiments“

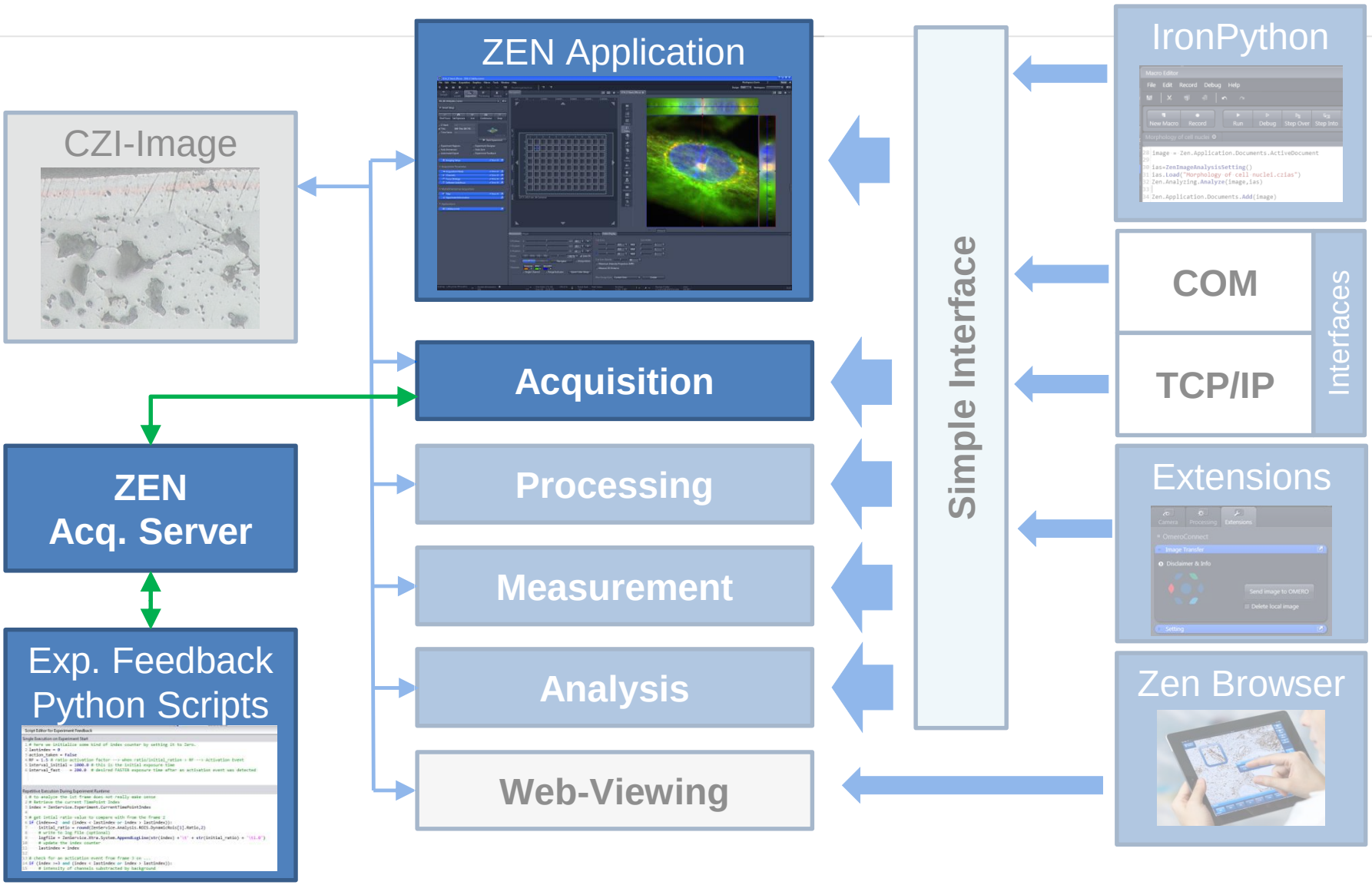
- Create dynamic acquisition-experiments
- Observe parameters during acquisition via online image analysis
- Monitor the status of the microscope and/or the sample
- Automatically react on changes of the sample or other parameters
- Modify the hardware/experiment parameters during the acquisition
  - increase integration time when the sample bleaches
  - stop the acquisition after a certain number of objects was detected

- Create custom log-files, integrate data logging in the ZEN experiment
- Start external applications (Python, Matlab,….)
- Display measurement results already during image acquisition

OAD – ZEN Interfaces



OAD – ZEN Interfaces



Experiment Feedback
ZEN Acquisition Server

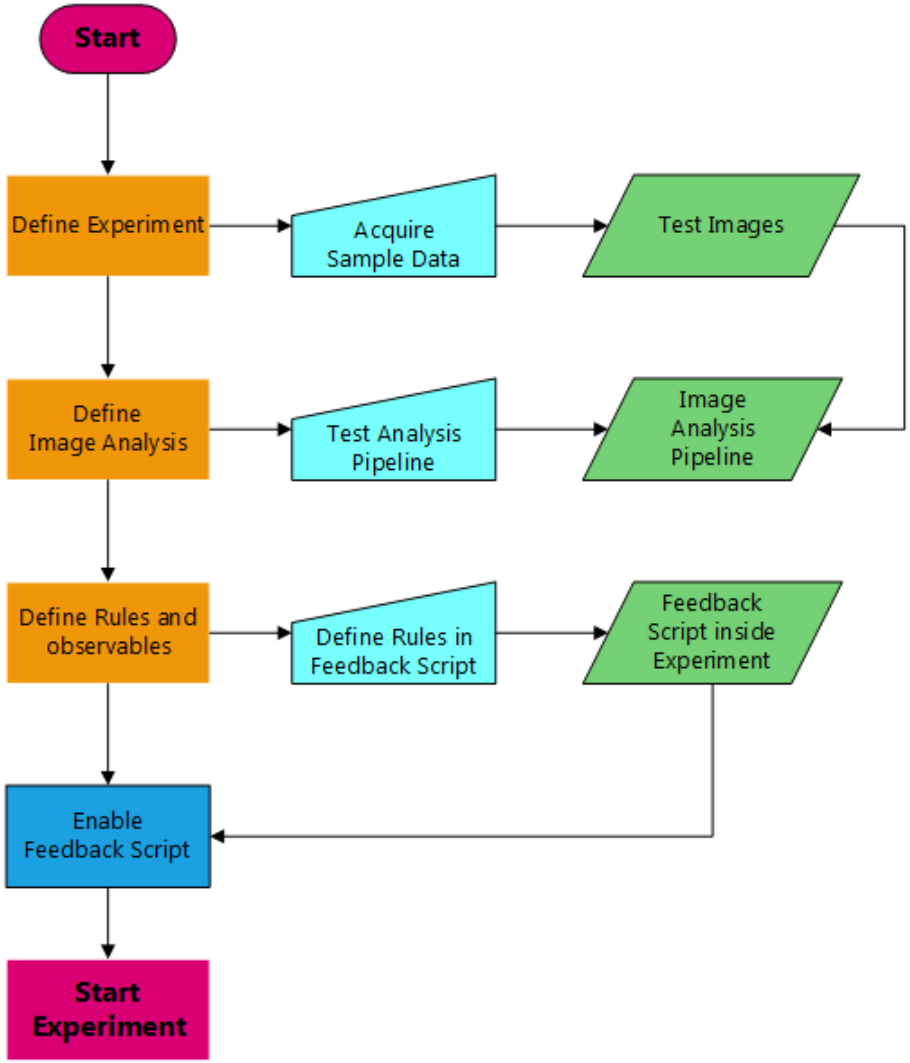
- **Adaptive Acquisition Engine**:

  Modify running experiments using Python scripts

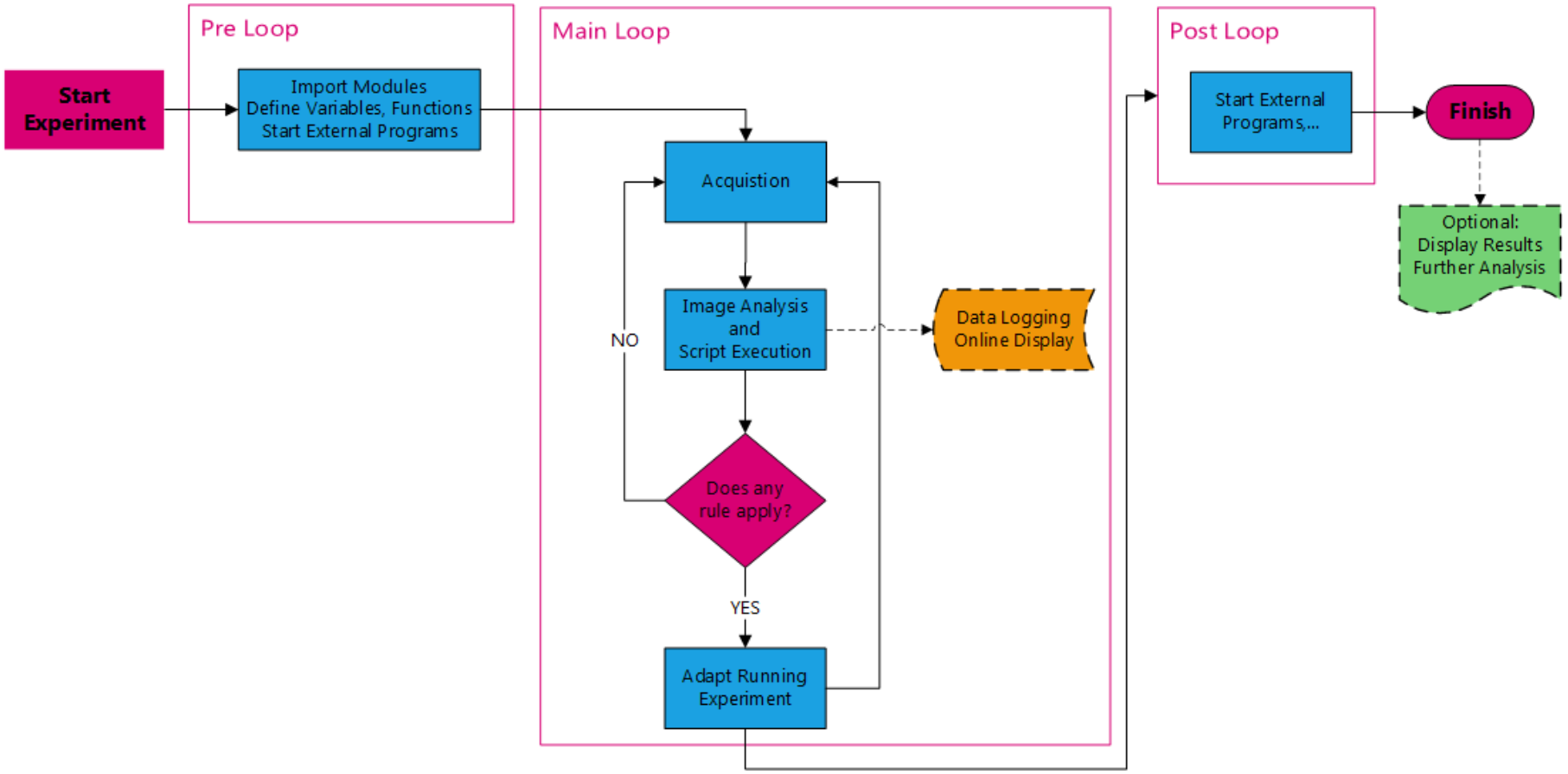
- Access the **current system status** & results from **online image analysis**

  on runtime during the experiment

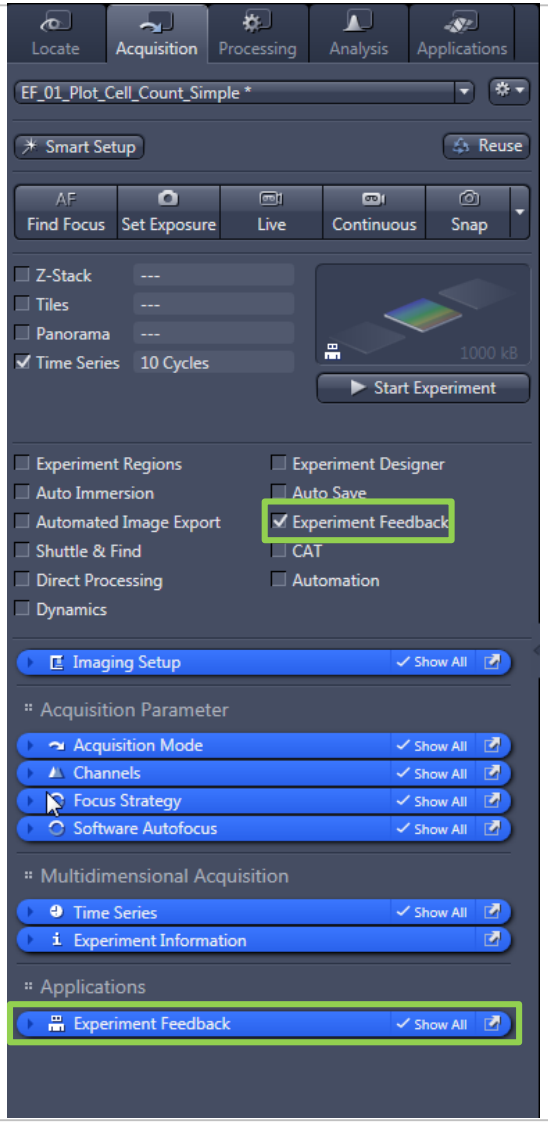
Experiment Feedback
Preparation

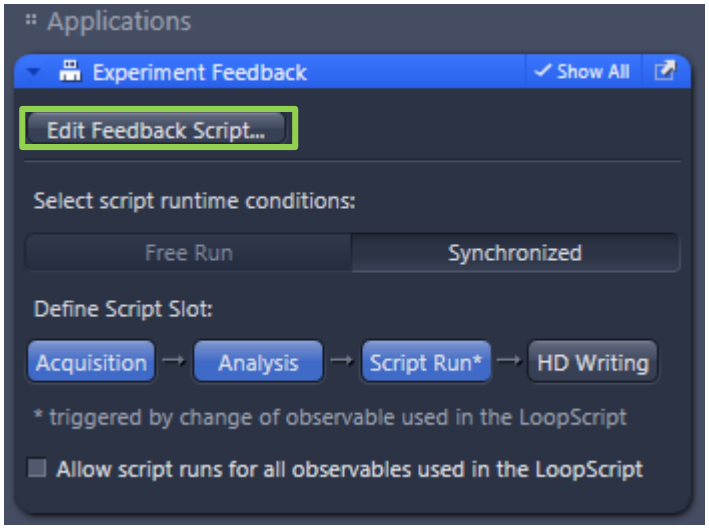


Experiment Feedback
Running the Experiment

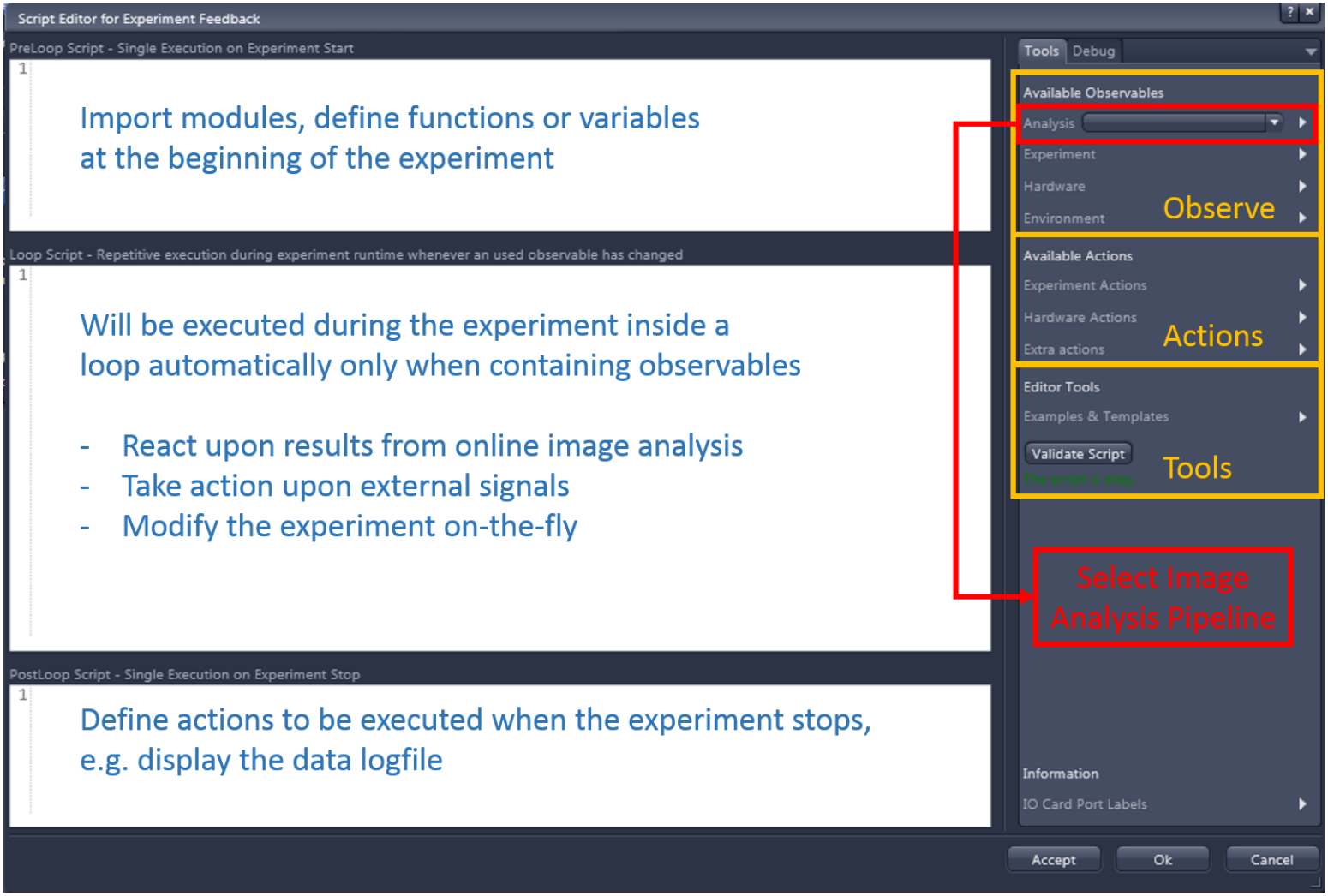


Create a Feedback Experiment

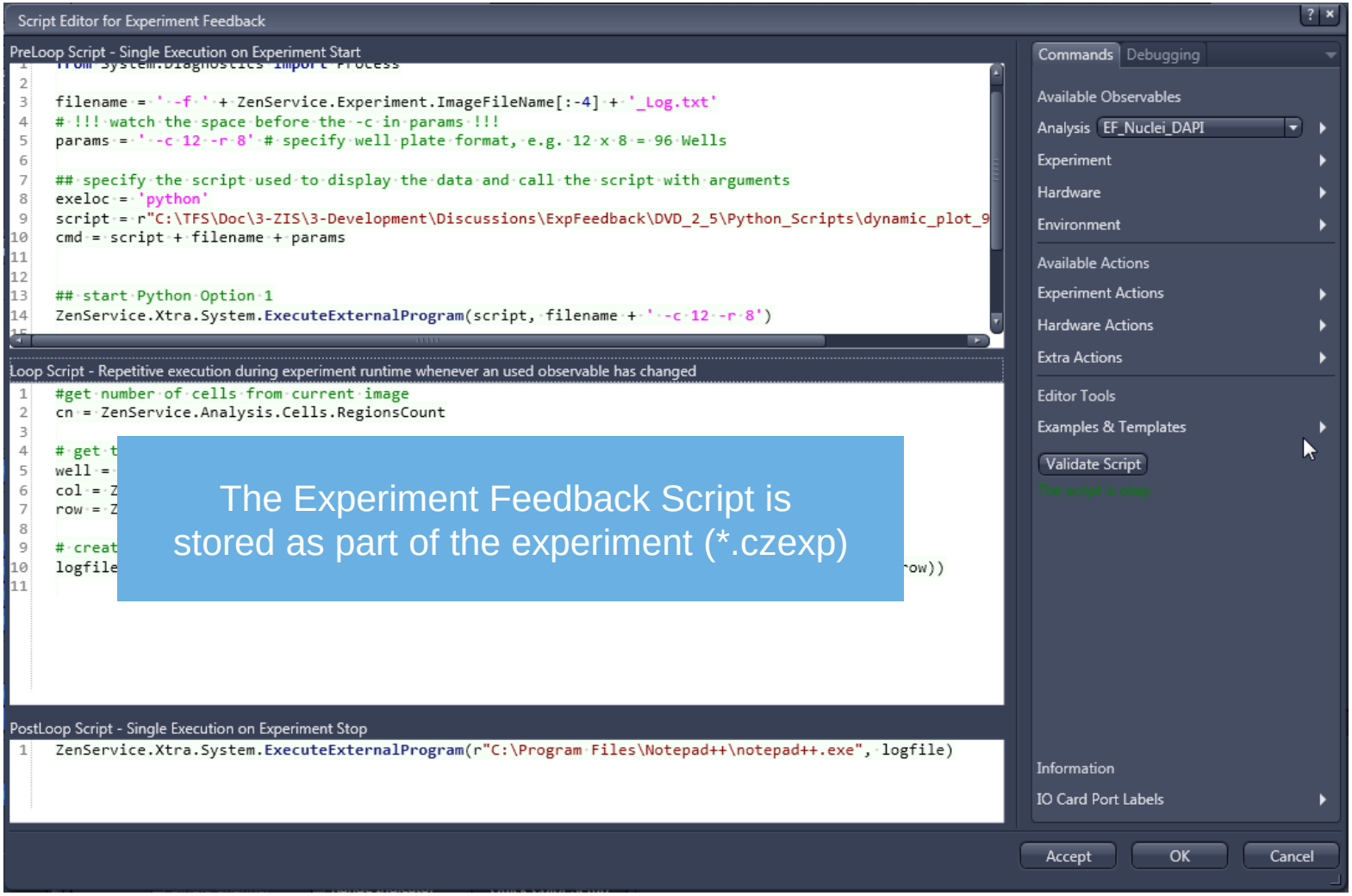




Experiment Feedback
Adaptive Acquisition Engine

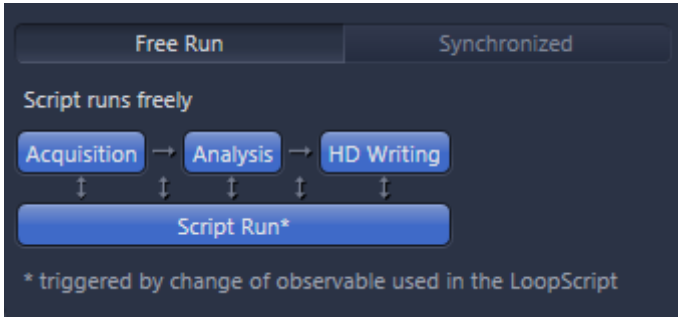


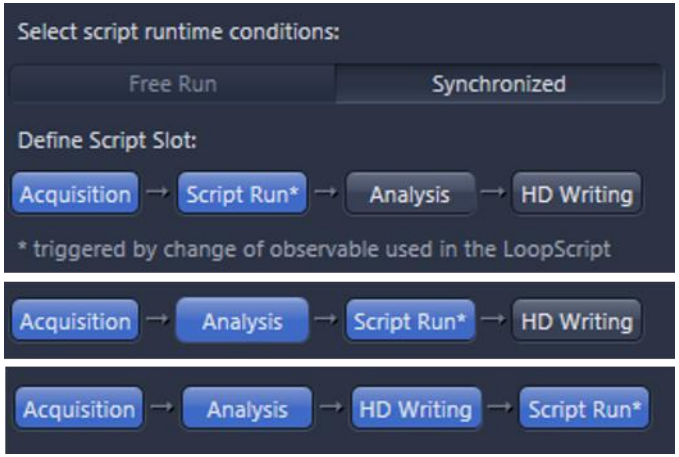
Experiment Feedback
Adaptive Acquisition Engine



Experiment Feedback: Run Options

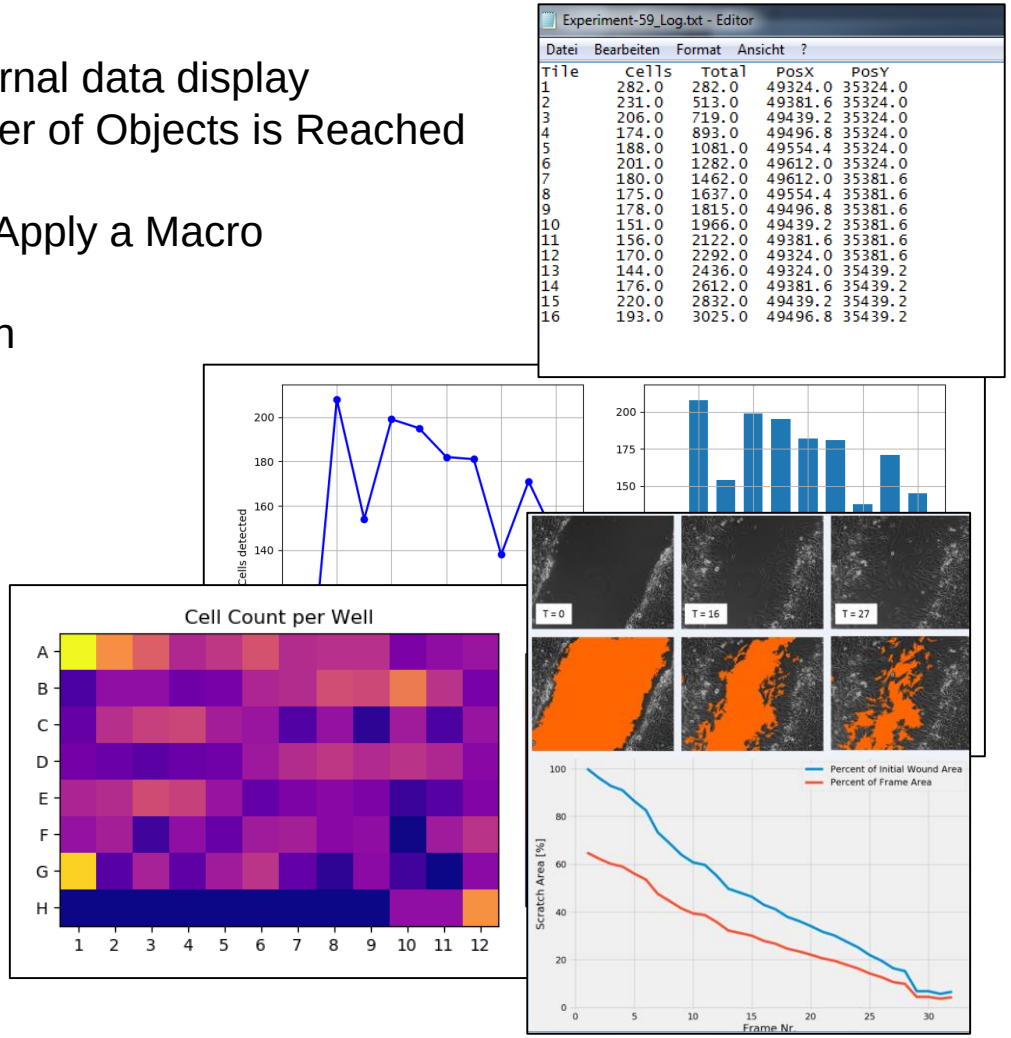
- Free Run
- Synchronized Run



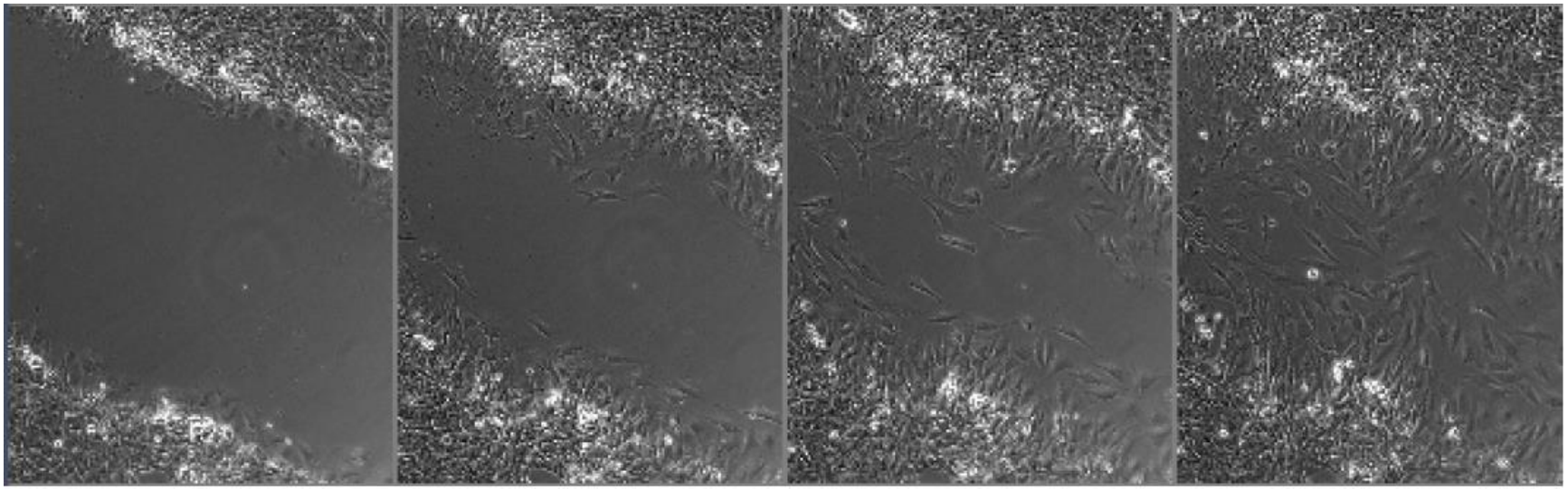


Sample Macros

- Count cells, write logfile and start external data display
- Acquire tile images until a Total Number of Objects is Reached
- Online data display
- Acquire Image Data, Open in Fiji and Apply a Macro
- Jump to next well
- Adapt exposure time during acquisition
- Modify the blocks of an experiment
- Time-lapse per Z-Plane
- Automatic event detection
- Online dynamics
- Online scratch assay
- Online tracking



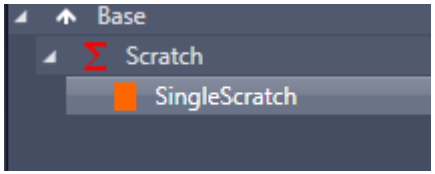
Online Scratch Assay Data

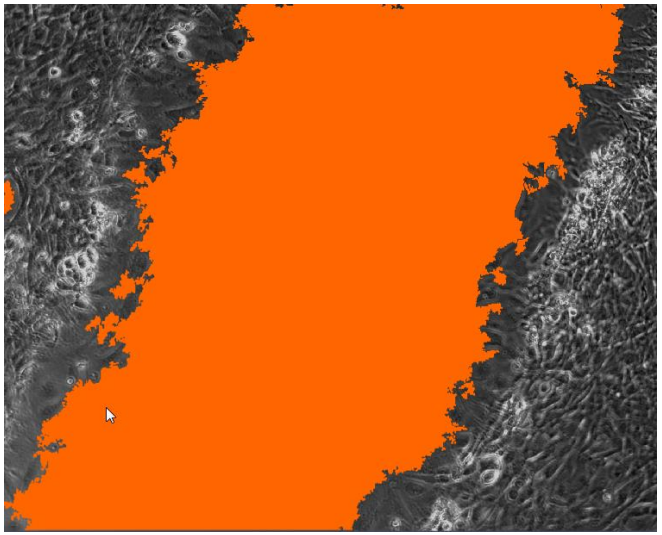


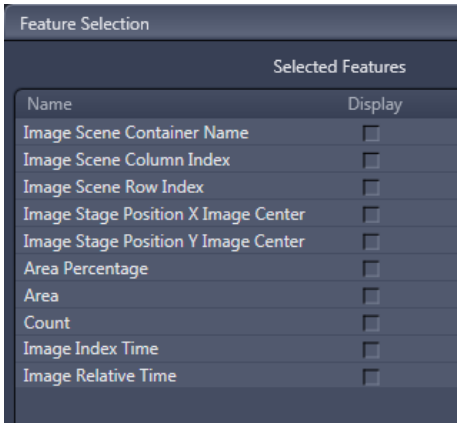
time

Online Scratch Assay
Image Analysis Setting

Classes → Segmentation → Features (Scratch)





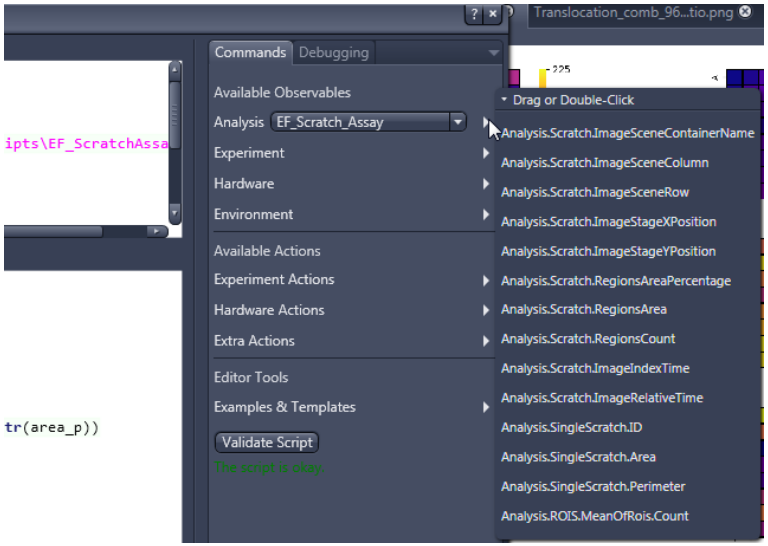


Online Scratch Assay
Pre-Script

```
from System.Diagnostics import Process

filename = ZenService.Experiment.ImageFileName[:-4] + '_Log.txt'

exeloc = 'python‘

script = r'C:\...\Python_Scripts\EF_ScratchAssay.py‘

cmd = script + ' -f ' + filename

## start Python Option 1
#ZenService.Xtra.System.ExecuteExternalProgram(script, ' -f ' + filename)

## start Python Option 2
app = Process();
app.StartInfo.FileName = exeloc
app.StartInfo.Arguments = cmd
app.Start()
```

Online Scratch Assay Loop-Script

```
# get the current well name, column index, row index and position index
frame = ZenService.Experiment.CurrentTimePointIndex

# get area parameters for the scratchnumber of cells from current image
area_t = ZenService.Analysis.Scratch.RegionsArea
area_p = ZenService.Analysis.Scratch.RegionsAreaPercentage

# create logfile
logfile = ZenService.Xtra.System.AppendLogLine(str(frame)+'\t'+str(area_t) + '\t'+
    str(area_p))
```



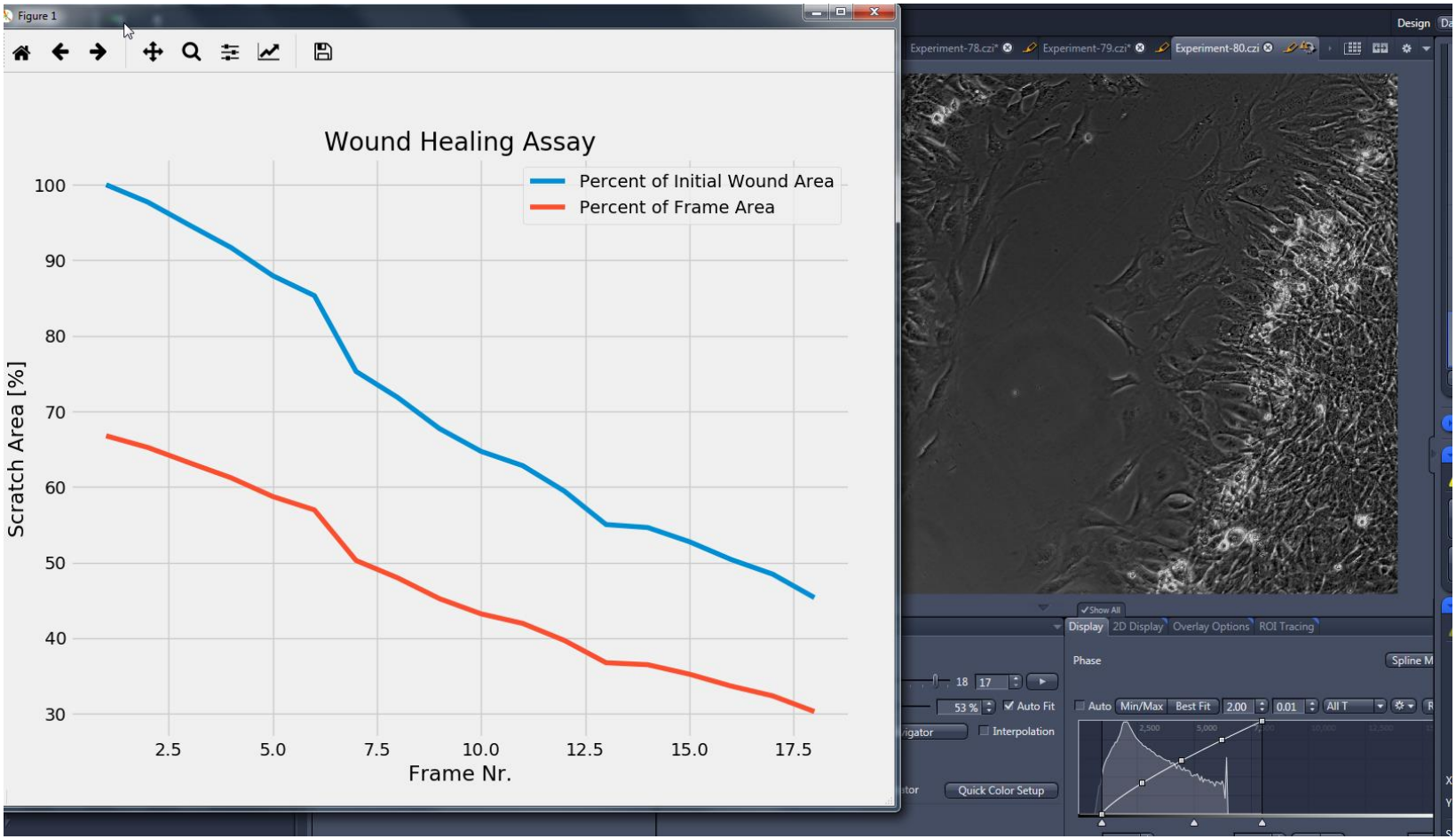
Online Scratch Assay
Python Script

```
import matplotlib.animation as animation
from matplotlib import style
import optparse
import numpy as np


# configure parsing option for command line usage
parser = optparse.OptionParser()

parser.add_option('-f', '--file',
    action="store", dest="filename",
    help="query string", default="No filename passed")

# read command line arguments
options, args = parser.parse_args()
savename = options.filename[:-4] + '.png'
print('Filename: ', options.filename)
print('Savename: ', savename)

# define plot layout
style.use('fivethirtyeight')
fig = plt.figure(figsize=(10,8))
ax1 = fig.add_subplot(1,1,1)
```

Online Scratch Assay
Python Script: Animate

```
def animate(i):

    try:
        graph_data = np.genfromtxt(options.filename, delimiter='\t')
        xs = graph_data[:,0] # get frame Nr.
        ys1 = graph_data[:,1] # get absolute Scratch area
        ys2 = graph_data[:,2] # get Sratch area in percent of Frame Area
        ys1_max = np.max(ys1, axis = 0) #get maximum Scratch Area
        ys1_percent = ys1/ys1_max*100   #normalize Scratch Area

        #labels and legend for plot
        ax1.clear()
        plt.title('Wound Healing Assay')
        plt.xlabel('Frame Nr.')
        plt.ylabel('Scratch Area [%]')
        ax1.plot(xs, ys1_percent, label = 'Percent of Initial Wound Area')
        ax1.plot(xs, ys2, label = 'Percent of Frame Area')
        ax1.legend(loc='upper right')

        #save plot
        plt.savefig(savename)

    except:
        print('No file loaded')

ani = animation.FuncAnimation(fig, animate, interval=1000, repeat=False)
plt.show()
```

Online Scatch Assay
Online Plotting



Online Scratch Assay
Post-Script

```
# open logfile
ZenService.Xtra.System.ExecuteExternalProgram(logfile, r'C:\...\notepad++.exe')
```

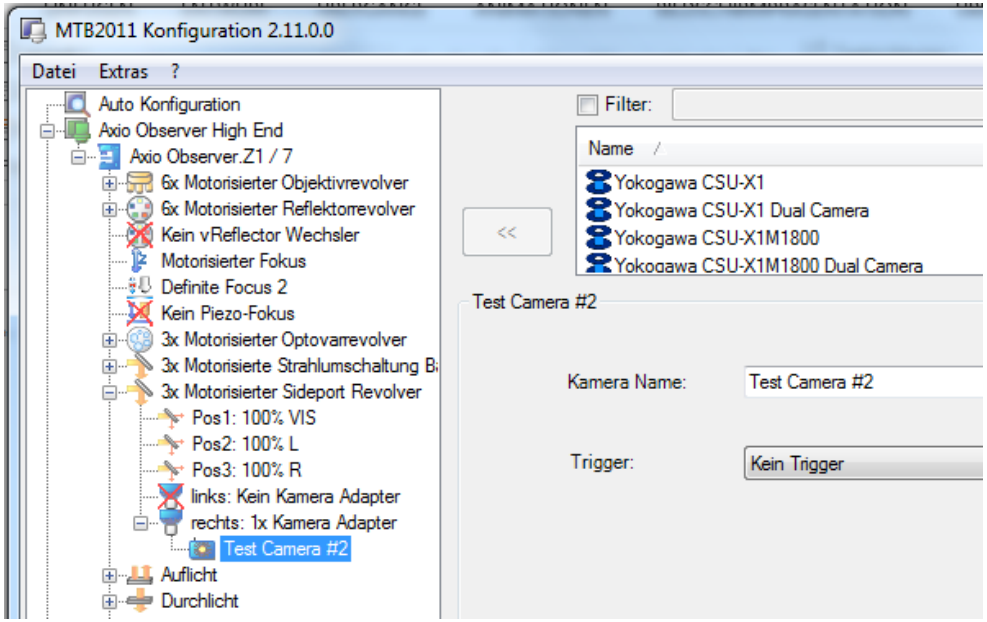
Experiment-79_Log.txt - Editor

Datei Bearbeiten Format Ansicht ?

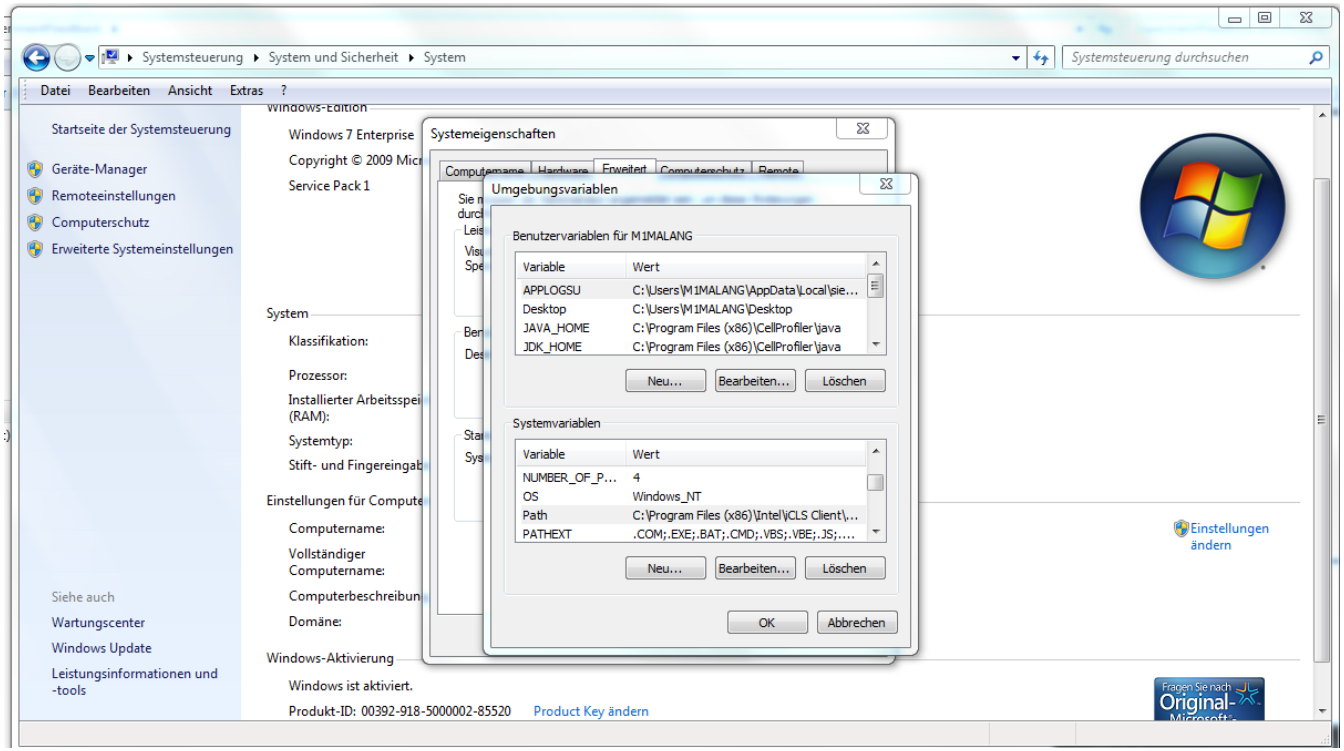
1	10143.73	66.7744275573
2	9910.62	65.2399045753
3	9601.87	63.2074565007
4	9299.96	61.2200349680
5	8923.39	58.7411395138
6	8654.6	56.9717412369
7	7638.16	50.2806917762
8	7287.65	47.9733448138
9	6871.32	45.2327161274
10	6565.3	43.2182391726
11	6373.17	41.9534804727
12	6030.98	39.7009026373

Copy **Active Configuration.xml** and **CZIS_Cameras.xml** to
C:\ProgramData\Carl Zeiss\MTB2011\2.11.0.0

Start MTB → Check “Simulate”
C:\Program Files\Carl Zeiss\MTB 2011 - 2.11.0.0\MTB Configuration\MTBConfig.exe
Check that there is a demo camera

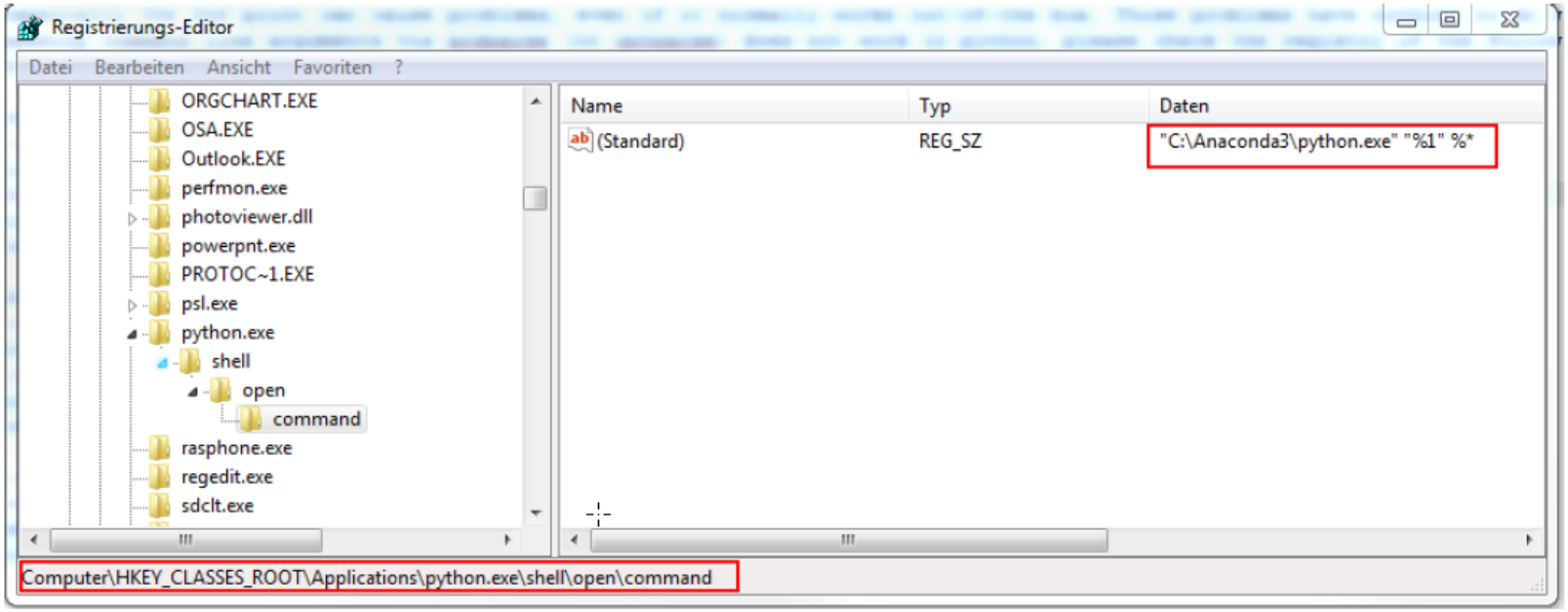


Add Python to the PATH variable

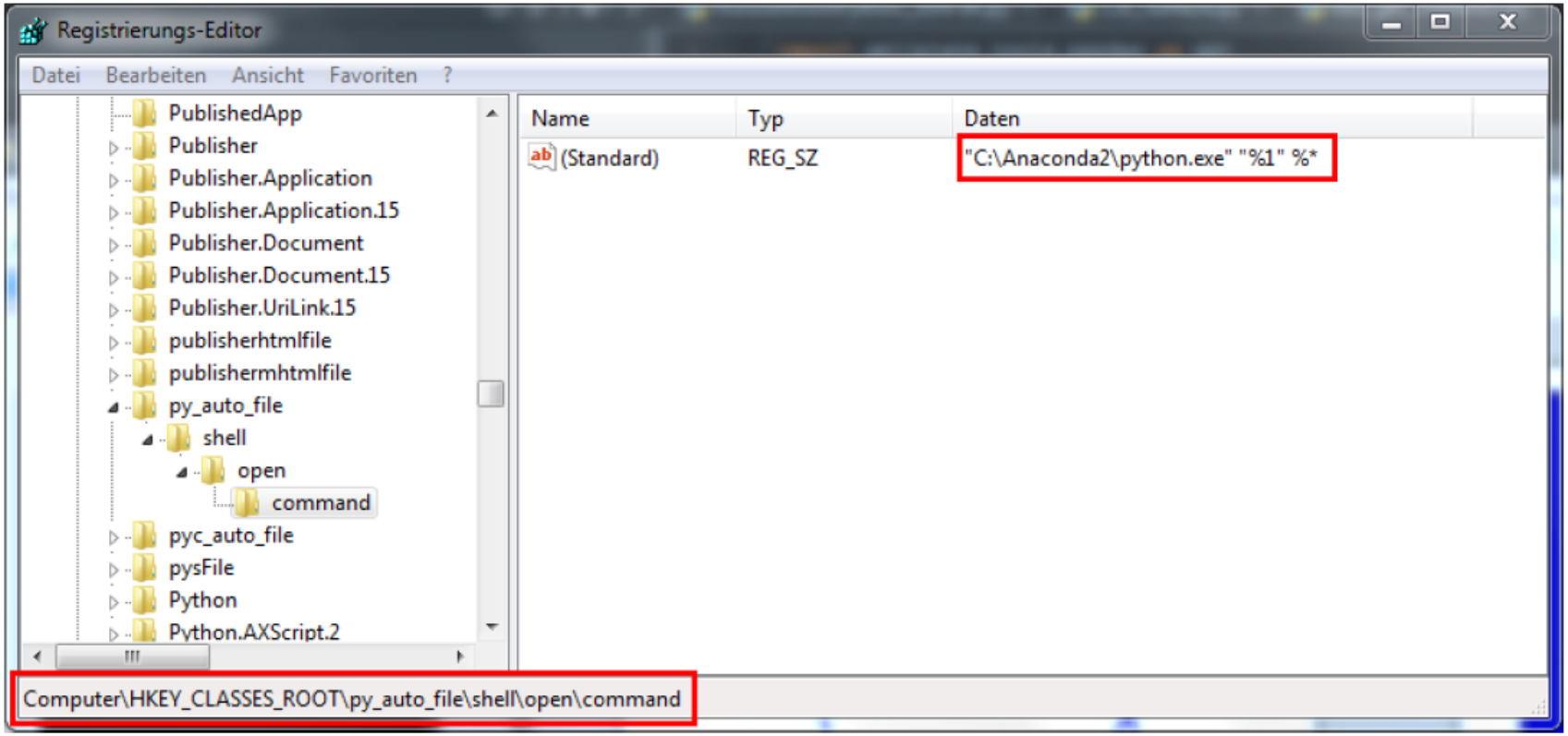


Alternatively: In the script specify the whole path to the python.exe

Registry


Registrierungs-Editor
Datei Bearbeiten Ansicht Favoriten ?
ORGCHART.EXE
OSA.EXE
Outlook.EXE
perfmon.exe
photoviewer.dll
powerpnt.exe
PROTOC~1.EXE
psl.exe
python.exe
shell
open
command
rasphone.exe
regedit.exe
sdclt.exe
Name
Typ
Daten
(Standard)
REG_SZ
"C:\Anaconda3\python.exe" "%1" %*
Computer\HKEY_CLASSES_ROOT\Applications\python.exe\shell\open\command

Registry


Registrierungs-Editor
Datei Bearbeiten Ansicht Favoriten ?
PublishedApp
Publisher
Publisher.Application
Publisher.Application.15
Publisher.Document
Publisher.Document.15
Publisher.UriLink.15
publisherhtmlfile
publishermhtmlfile
py_auto_file
shell
open
command
pyc_auto_file
pysFile
Python
Python.AXScript.2
Name
Typ
Daten
(Standard)
REG_SZ
"C:\Anaconda2\python.exe" "%1" %*
Computer\HKEY_CLASSES_ROOT\py_auto_file\shell\open\command

Copy Demo

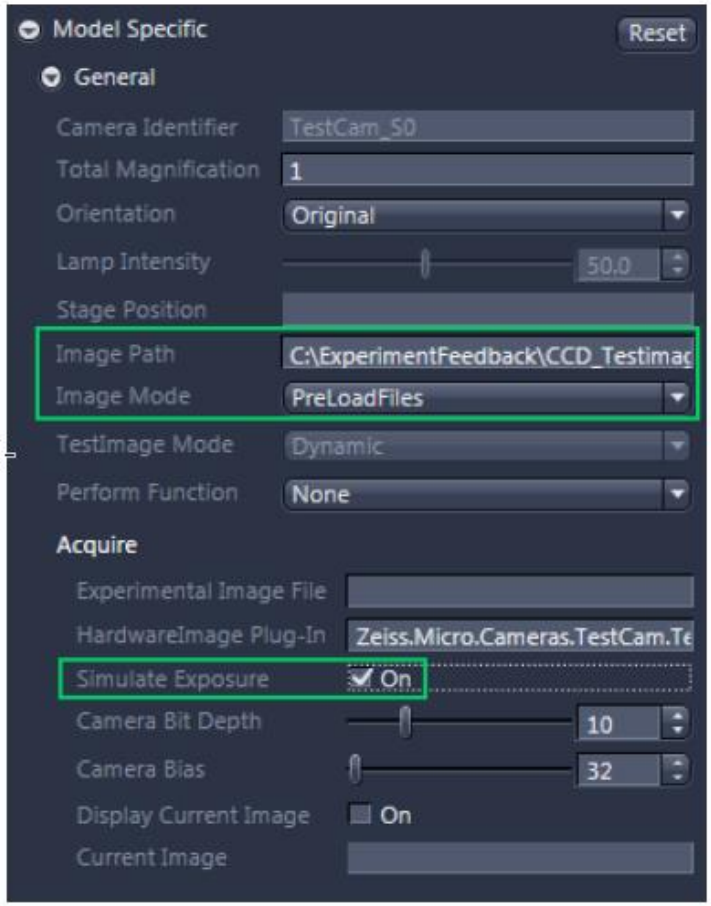
To C:\Users\YourUsername\Documents\Carl Zeiss\ZEN\Documents

- Content of „Exp with Scripts“ → Experiment Setups
- Content of „Image Analysis“ → Image Analysis Settings

To a local folder:

- Test Images
- Python Scripts

Simulate the Acquisition




ZEISS