Custom Module Documentation

CONTENT OVERVIEW:

- Introduction
- Hosted model script
- Best practices for writing scripts with custom library
- Limitations

INTRODUCTION:

This document is prepared in the way to convey that we have a method to include custom library files to make it work with the Endpoint manager i.e through **Script**. Customer don't need to be a script writer, we make our customer's to develop a new script to import the modules they want to work with. So far, we are supporting all Python 2.7.13 standard libraries addition with the modules listed below by default.

gyp-0.1
markerlib
PyZDDE
tcl8
tcl8.5
tix 8.4.3
tk8.5
reg1.2 (tclreg12.dll)
pip-7.1.2
setuptoost-18.2

Our embedded library files are available at "**HOMEPATH\COMODO\Comodo ITSM\Lib\site-packages**" in the endpoint once Endpoint Manager agent is installed.

HOSTED MODEL SCRIPT:

Here is an example script link for the custom module implementation:
https://scripts.comodo.com/frontend/web/topic/custom-module-implementation

BEST PRACTICES FOR WRITING SCRIPT WITH CUSTOM LIBRARY:

To import and avail custom library packages with the Endpoint manager, we offer a way to our customer to write the scripts on own to experience the performances instead of achieving through manual operations or through Endpoint manager. We feel proud to make our customer to be a code writer. Let's join your hands and walk through with below mentioned hits:

- Host a complete module with all relevant packages as zip file.
- Download the hosted module on the Endpoint.
- Extract the zip and copy the matters.
- Import the module.

HOST A MODULE:

It's always a good deal to host any matters on any space and get it anywhere by using the custom URL. With that hope we hosted a module as an example for your view to perform the same.

In the above displayed script we have used an hosted URL (https://drive.google.com/uc?export=download&id=1h3nvQ7hRD68FeRsP5qWjEi1phHAeiL5L) which has module called "**PATHLIB**" as zip file.

DOWNLOAD MODULE:

Python allows an easy method to download from custom URL using magic browser. This browser eliminates user interactions, suppresses UAC prompt etc. The download function takes arguments as **URL** and **FileName.** It reads a file present in the URL, writes the same matters by opening a file in the Endpoint with the same extension. Another choice from our Endpoint manager is, download action can be performed as Administrator or Logged in user.

Supported modules are: **urllib** & **ssl**

Below content in the script is to download the hosted module.

Download function

```
################################################
###         Download Hosted Module           ###
################################################

def Download(URL, DownloadTo = None, FileName = None):
    import urllib
    import ssl
    if FileName:
        FileName = FileName
    else:
        FileName = URL.split('/')[-1]

    if DownloadTo:
        DownloadTo = DownloadTo
    else:
        DownloadTo = os.path.join(os.environ['TEMP'])

    DF = os.path.join(DownloadTo, FileName)
    with open(os.path.join(DownloadTo, FileName), 'wb') as f:
        try:
            context = ssl._create_unverified_context()
            f.write(urllib.urlopen(URL,context=context).read())
        except:
            f.write(urllib.urlopen(URL).read())
    if os.path.isfile(DF):
        return DF
    else:
        return False
```

EXTRACT ZIP:

We do advise our customer that the hosted module package should be hosted as zip file. Since, zip performs complete compress methods to avoid files or packages escape from the regular space.

Supported modules are: **zipfile**

Below content is to extract the zip file at the locations: "**HOMEPATH\COMODO\Comodo ITSM\Lib\site-packages**".

ZIP

```
################################################
###          Zip downloaded Module           ###
################################################

def zip_item(path,final_path):  # Creating ZIP file
    import zipfile
    zip_ref = zipfile.ZipFile(path, 'r')
    zip_ref.extractall(final_path)
    zip_ref.close()
    return final_path
```

IMPORT MODULE:

Before step to import the downloaded and extracted module, we should select the OS of an Endpoint to occupy a valid path. Once, path is selected an **import module** will be called which has **download** and **extract** functions integrated to it and copy the extracted file to the Endpoint manager location.

Below content in the script is to Import the hosted module.

Import

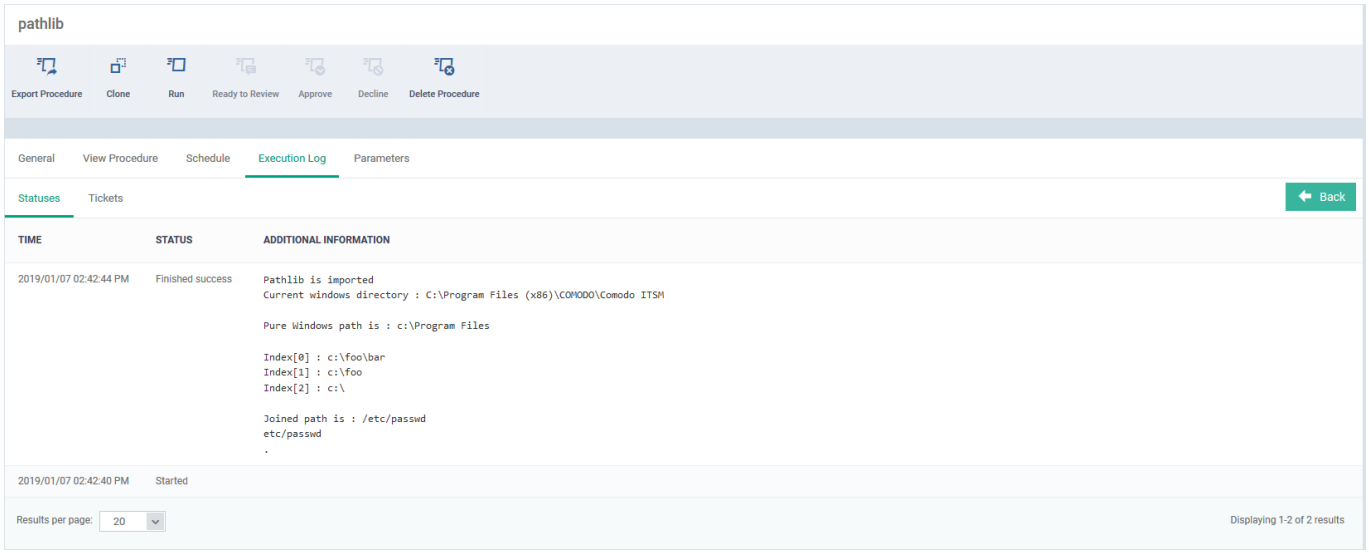
```
################################################
###           Place zipped Module             ###
################################################

def Import_module(DEST):
    BDPATH = Download(r'https://drive.google.com/uc?export=download&id=1h3nvQ7hRD68FeRsP5qWjEi1phHAeiL5L', FileName = 'pathlib.zip')
    SRC = os.path.join(os.environ['TEMP'])
    path=zip_item(BDPATH,SRC)
    SRC = os.path.join(os.environ['TEMP'],'pathlib')
    from distutils.dir_util import copy_tree
    copy_tree(SRC, DEST)

    import pathlib
    print "Pathlib is imported"


    return DEST

################################################
### Find 32 or 64 bit & set path for libraries ###
################################################

HOMEPATH = r"C:\Program Files (x86)"
if os.path.exists(HOMEPATH):
        HOMEPATH = r"C:\Program Files (x86)"
else:
    HOMEPATH =r"C:\Program Files"

DEST= os.path.join(HOMEPATH,r'COMODO\Comodo ITSM\Lib\site-packages')
DEST=Import_module(DEST)
```

Example module which has been demonstrated in this script is "**PATHLIB**"

pathlib

```
import pathlib
from pathlib import *

# Current windows directory
c=Path.cwd()
print "Current windows directory : "+str(c)+"\n"

# Make it pure windows path
print "Pure Windows path is : "+str(PureWindowsPath('c:/Windows', '/Program Files'))+"\n"


# Retrieve paths per index
p = PureWindowsPath('c:/foo/bar/setup.py')
print "Index[0] : "+str(p.parents[0])
print "Index[1] : "+str(p.parents[1])
print "Index[2] : "+str(p.parents[2])


# join Path
print "\n"+"Joined path is : "+str(PurePosixPath('/etc').joinpath('passwd'))


# Split from first
p = PurePosixPath('/etc/passwd')
print p.relative_to('/')
print p.relative_to('/etc/passwd')
```

RESULT CASE for above example script.



LIMITATIONS:

It is important to know that the method we are explaining here to support custom python library would work only if the corresponding library doesn't have dependency problem with other library files.

Ensure that the module doen't initiate UAC prompt to need user input.

Thus this work flawless only if the module support this method.