How to write scripts for WinRM + AD Connector

This page main purpose is to inform developers how to write python and powershell script for our WinRM + AD Connector

Before you start with scripts it's good to know how this connector is working and what is his purpose so please follow the link above a read some information about the connector.

Python

The best way how to show what's the basic logic of python scripts is to show it at some simple example script with additional comments.

This is example create script

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# first line is just shebang line and second is for defining Python source code encodings
# File should be in UTF-8
# Be careful about line ending. If you are using connector server on Linux environment be sure that python script has LF ending otherwise python will not be executed

# All params from IdM is stored in environment of this script and you can get them by os.environ["paramName"]
# For getting attribute you can use prepared method winrm_wrapper.getParam("paramName") which has the benefit of returning unicode value and in case the attribute is not existing
# in environment it will not fail but return empty string

import sys, os
# this is needed for importing file winrm_wrapper from parent dir
sys.path.append(os.path.join(os.path.dirname(__file__), '..'))
import winrm_wrapper
import json

# Getting attribute which is marked as identifier
uid = winrm_wrapper.getParam("sAMAccountName")

# For logging you can use prepared method from wrapper
winrm_wrapper.writeLog("Create start for " + uid)

# Parse multi valued attribute to list
if winrm_wrapper.getParam("ldapGroups"):
```

```
    roles = json.loads(winrm_wrapper.getParam("ldapGroups"))

# Load PS script from file and replace params
winrm_wrapper.writeLog("loading script")
f = open(os.environ["script"], "r")
command = f.read()
# this means that we replace $uid in powershell script with out uid.
# unfortunately there is no better solution how to send params with
powershell
# that;s the reason why we need to replace them like this, because we need
to send complete command
command = command.replace("$uid", uid)

# Call wrapper
winrm_wrapper.executeScript(os.environ["endpoint"],
os.environ["authentication"], os.environ["user"],
                                    os.environ["password"], command, uid)

winrm_wrapper.writeLog("Create end for " + uid)
print("__UID__=" + uid)
sys.exit()
```

Powershell

When connector server is on Windows use Write-Output instead of Write-Host

I'll use same solution as with Python script and jump directly to some example script

Tips:

- Use -Confirm:$false parameter to avoid "freezing" of your script
- Use -ErrorAction Stop or -ea Stop for better error handeling, because some command will print error to stdout by default so you won't be able to catch them without this parameter

This is example search script so I can show handling of response

```
# Search script, which will return information about user's exchange account
# INPUT:
#   uid - String - account identifier

Write-Host "PS Search started"

#Needed to load Exchange cmdlets
Add-PSSnapin -Name '*Exchange*'

# Wrap logic in try catch. If you not handle errors correctly then IdM will
```

```
has no chance how to know if there was error or no. So best practice is to
return exit 0 if everything is ok
# return some other code in case of error
# Write error messages to stderr
try {
    #$uid will be replace with some value from python, in case that search
is for all (reconcilation) we will have empty string here that's the reason
why we are assigning the value to new variable

    $identificator = "$uid"
    $obj

    # check if identifier is empty or not
    if ([string]::IsNullOrEmpty($identificator)) {
        # save command output to variable
        $obj = Get-RemoteMailbox
    }
    else {
        # save command output to variable
        $obj = Get-RemoteMailbox -Identity $identificator
    }

    # parsing properties for IdM
    # IdM will accept JSON which is List of Maps where key is name attribute
and value is value

    # prepare list

    $resultList = @()

    # Iterate thru response object e.g. We get 10 users so we need to create
map for each of them inside this loop and add it to list
    foreach ($item in $obj) {

        # prepare map
        $resultMap = @{ }

        # iterate thru each result attributes
        foreach ($attr in $item.psobject.Properties) {

            # care only about attributes which has some value
            if (![string]::IsNullOrWhitespace($attr.Value)) {
                $name = $attr.Name
                $value = $attr.Value

                $resultMap.add("$name", "$value")

                # now we need to fill __UID__ and __NAME__ attribute as
connid needs this values
                if ($name -eq "SamAccountName") {
                    $name = "__UID__"
```

```
                    $resultMap.add("$name", "$value")
                    $name = "__NAME__"
                    $resultMap.add("$name", "$value")
                }
            }
        }
        $resultList += $resultMap
    }
    # convert to json
    ConvertTo-Json $resultList
}
catch {
    # Write to stderr and exit with code 1
    [Console]::Error.WriteLine($_.Exception)
    exit 1
}
Write-Host "PS Search ended"
```