Powershell Password Change Job for IPMI Devices

Start with the following PowerShell snippet to create a password change job for an IPMI device that has been *previously enrolled*.

Note that there is only a cmdlet to perform a Windows password change job. All jobs start as Windows password change jobs and are transformed into jobs for the other target platforms. This the proper way to create a password change job for an IPMI device is:

1. New-LSJobWindowsChangePassword - this creates the initial job (yes, for the wrong platform).
2. Set-LSJobChangePasswordChangeSettings - this will let you define everything, including that you want to do it for an IPMI device.
3. Set-LSJobSchedule

In the example code below, the schedule portion of the job is not configured.

```
#The following will create an IPMI password change job that will login as ADMIN using the
password from the store and randomize an account called Scooby.
#Be aware, most of these devices are case sensitive!!!
#You have to authenticate first!

$token = Get-LSLoginToken

#Set target elements $tsystem = '192.168.1.151'

$lAccount = 'ADMIN'
#$lAPwd = 'ADMIN' #shouldn't be needed as the device is already enrolled with a password as a
valid account store. But just in case, use this value when 'UseSavedPasswords = 0' for
JobSettings
$tType = 'ACCOUNT_TYPE_IPMI' #This sets the job to an IPMI password change job
$tAccount = 'Scooby'
$pwCType = 'PWD_CHANGE_TYPE_GEN_RANDOM' #This sets the job to use a random password rather than a
fixed/static password
$pwCompat = 'PWD_COMPAT_W2K' #This sets the job to use WIN2K compatibility mode so you can use
passwords longer than 14 characters

#Create the new job

#The new job is added to a variable $nJob as we will need to pull the JobID from the message
output

$nJob = New-LSJobWindowsChangePassword -SystemName $tsystem -AccountName $tAccount -
PasswordLength 10 -AuthenticationToken $token -CreateAccountIfNotFound $False

#Grab JobID from Operation Message. JobID is the last element in the output message on success.

$OMArray=$nJob.OperationMessage.Split()
$JID = $OMArray[-1]

#Update the job to be used for IPMI rather than for Windows and set other password constraints

#Create PCJobSettings Objects

$PCJobSettings = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.PasswordChangeSettings
$PWConstraints = New-Object -TypeName
```

```
LSClientAgentCommandlets.RouletteWebService.PasswordChangeConstraints

#Define job settings

$PCJobSettings.AccountType = [LSClientAgentCommandlets.RouletteWebService.EAccountType]::$tType
$PCJobSettings.PasswordChangeType =
[LSClientAgentCommandlets.RouletteWebService.EPasswordChangeType]::$pwCType
$PCJobSettings.PasswordCompatibilityLevel =
[LSClientAgentCommandlets.RouletteWebService.EPasswordCompatibilityLevel]::$pwCompat
$PCJobSettings.FullAccountName = $tAccount
$PCJobSettings.LoginName = $lAccount
#$PCJobSettings.CurrentPassword = $lAPwd #Uncomment when 'UseSavedPasswords = 0'
$PCJobSettings.Unique = 1 #Set to 1 to use a unique random password rather than a fixed random
password
$PCJobSettings.UseSavedPasswords = 1 #Set to 1 to instruct the job to use the password stored for
the login account. If set to 0 you must supply (uncomment) the Current password and #lAPwd above
$PCJobSettings.FirstCharacterSetBits = 15 #This sets the password settings to use all possible
characters in the first position
$PCJobSettings.LastCharacterSetBits = 15 #This sets the password settings to use all possible
characters in the last position
$PCJobSettings.MiddleCharactersSetBits = 15 #This sets the password settings to use all possible
characters in the middle position
$PCJobSettings.PasswordCharacterSetBits = 15 #This sets the password settings to use all possible
characters for the general job
$PCJobSettings.PasswordLength = 8
$PCJobSettings.MinLettersLcase = 1
$PCJobSettings.MinLettersUcase = 1
$PCJobSettings.MinNumbers = 1
$PCJobSettings.MinSymbols = 1
$PCJobSettings.PasswordSegments = 1

#Define constraints

$PWConstraints.FailGenerationOnMissingPassfiltDLL = $False #If set to $True or not included, you
must also define a path to the password filter or the job will fail.
$PWConstraints.SymbolsExcludeProblematicWithAPIs = $True #Set to $False to enable all possible
characters in the job

#Assign the password constraints

$PCJobSettings.PasswordConstraints = $PWConstraints

#Apply the job with the new settings

Set-LSJobPasswordChangeSettings -AuthenticationToken $token -PasswordChangeSettings

$PCJobSettings -JobId $JID

#Define the Job Schedule (commented out in this example as it is not configured)

#Set-LSJobSchedule
```