

Some Scripting for the MSI/GPO installations

Background

So, you've created an MSI installer and created a policy to push that installation down to computers in your OU. Only one hurdle remains; you must reboot all of the systems for a machine-level software installation to take place! And you thought you were going to get away with not touching those computers. Well, maybe you still can...

The Process

If only you had a list of the computers in your OU. Wait! The OU contents are a list. So we just need to:

1. Extract a list of computers in the OU
2. Format and store the list
3. Run a remote reboot on those systems
4. (optional) Run secedit on those computers

That last part is really only important if you've made accommodations for the new software in your security policy (typically, file system permission changes made via GPO).

The Tools

Some software is needed to successfully complete the scripting. Here's what I use:

1. LDIFDE.EXE – Used to query LDAP objects, including OUs. I'll use this to get this list of computers in the target OU. This file exists on Windows 2000 Servers. Copying the executable to a Windows 2000 Professional workstation works fine.
2. PSShutdown.EXE – Used to remotely reboot the computers in the OU. You can download this as part of the PSTools 1.8 package from www.sysinternals.com.
3. PSEXEC.EXE – Used to remotely execute commands on Windows computers. This is also part of the PSTools 1.8 suite.
4. Perl – I can't write a script without a language to write it in! I'm choosing Perl because it's a good tool for just such a job, I'm pretty familiar with it, and it's free. You should be able to adapt the code snippets to almost any language you desire. The version I'm using is ActivePerl 521, which is on the Windows 2000 Server Resource Kit. Newer versions are available for download from www.activestate.com, and should work just fine.

Step-By-Step

I'll write one script to get my list of computers, then format and store that list. The heart of this is the LDIFDE program, which can get me a nice list of the objects I request. The format of the LDIFDE command looks like this:

```
LDIFDE.EXE -f ldifdump.txt -s FREEDOM -d
"OU=0_test,OU=central,dc=CNTRLSRVS,dc=w2k,dc=vt,dc=edu" -p
subtree -r "(ObjectClass=Computer)" -l "SamAccountName"
```

Let's break that down a bit:

- The `-f` directive tells LDIFDE what file name to use to store the results.
- The `-s` directive tells LDIFDE what computer to contact (in this case, FREEDOM is one of the Central Services domain controllers).
- The `-d` parameter specifies the container to query. In this case, we're querying the Central\0_test container in the cntrlsrvs.w2k.vt.edu domain. Please note that the OUs are listed in reverse order.
- The `-p` directive tells LDIFDE the scope of the query, a "subtree" query here. If we have sub-OUs (which we don't) and wanted to include the contents of those as well (normally when pushing a policy, you would), subtree is a great choice.
- The `-r` option used here returns only computer objects to us, thus we don't try to reboot any user or group accounts.
- The `-l` option tells LDIFDE what information to return. In this case, we just want the computer name, so we'll take the SamAccountName (this includes the trailing \$ symbol that normally comes with computer accounts).

The LDIFDE output looks like this:

```
dn: CN=SPPUSHTEST,OU=0_Test,OU=Central,DC=cntrlsrvs,DC=w2k,DC=vt,DC=edu
changetype: add
sAMAccountName: SPPUSHTEST$
```

Here, there's just one computer in the OU at the time. The file is formatted such that it can be run back through LDIFDE using the `-i` parameter to be imported. The extra information is not required for our uses, and we'll ignore it.

The script will run this command, then open the ldifdump.txt file, read the relevant entries, and output a neatly-formatted, one-per-line file with a list of the computer accounts. The parameters are variables in the script, so one can just edit the script slightly to change them. And here's the script:

```
#getcomps.pl v1.0 - Mike Moyer, Virginia Tech, 2003
#enumerates computer accounts from an AD OU via LDAP query
#ldifde.exe must be in the path, or in the directory
#(get from W2k server if on W2k Pro)

#path to ldifde.exe program, if not in path/directory
#include trailing \\
$path = "";

#temporary output file (existing will be overwritten)
$f = "temp.txt";
```

```

#formatted output file
$c = "compacts.txt";

#server name to contact
$s = "FREEDOM";

#OU to query, e.g. to query vt\mydept, use OU=mydept,OU=vt
$ou = "OU=0_test,OU=central";

#Domain container of OU, e.g. cntrlsvs.w2k.vt.edu, use
#dc=cntrlsvs,dc=w2k,dc=vt,dc=edu
$dc = "dc=CNTRLSRVS,dc=w2k,dc=vt,dc=edu";

#Query type, subtree|onelevel|base, probably want subtree
$p = "subtree";

#ObjectClass, we're looking for computers, so (ObjectClass=Computer)
$r = "(objectClass=Computer)";

#What we want, in this case the SamAccountName
#Which is the Computer Account Name
#Separate multiples with commas
$l = "SamAccountName";

#Account credentials, "user domain password"
#You probably don't want to use this, just log in with a proper
#account before you run the script
#$b = 'myacct mydomain mypassword';

$Command = $path."ldifde.exe -f $f -s $s -d \"$ou,$dc\" -p $p -r \"$r\"
-l \"$l\"";
if ($b) {$Command .= " -b $b";}
`$Command`; #backticks, not single-quotes (see Perl process mgmt)
open (RAWLIST, "<$f") or die "unable to open temp work file: $!\n";
open (OUTPUT, ">$c") or die "unable to open output file: $!\n";
while (<RAWLIST>)
{
    #Strip out the lead text
    if (/sAMAccountName\: (.*)\$/i) {print OUTPUT "$l\n"};
}
close (OUTPUT);
close (RAWLIST);

#Now the file from $c should contain all the computer names
#in our target OU, one per line, excluding the trailing $

```

Now we have a list of computers, and we need to remotely reboot them. Enter this small script to run PSSHUTDOWN.EXE to reboot all the machines. Note that you must be logged in to the computer from which you're running this script with an administrator-privileged account on each of the remote systems (pssshutdown and psexec claim command line options for substituting user names and passwords, but they don't seem to work):

```
#Options -r=reboot, -f=force (no save, warn your users)
```

```

#others would be -t # (delay in seconds)
$o = "-r -f";

#Path, if pssshutdown.exe is not in the path or directory
#include trailing \, please
$path = "";

#File to get computer accounts from
$c = "compacts.txt";

open (LIST, "<$c") or die ("unable to open $c: $!\n");
open (LOG, ">rebootem.log") or die ("unable to open log file: $!\n");
foreach (<LIST>)
{
    next unless ($_);
    $Command = $path."pssshutdown.exe $o \\\$_";
    print LOG "$Command\n";
    @output = `"$Command"`; #backticks
    print LOG @output;
}
close (LOG);
close (LIST);

```

Pretty simple stuff. Now, if we need to run SECEDIT (because we have some other part of the policy that was “enforced” before the software was installed, and we need to have it “re-enforced”, and we don’t want to wait for the next refresh [90 minutes, default]):

```

#Remote command to run
$o = "secedit /refreshpolicy MACHINE_POLICY /enforce";

#Path, if pssshutdown.exe is not in the path or directory
#include trailing \, please
$path = "";

#File to get computer accounts from
$c = "compacts.txt";

open (LIST, "<$c") or die ("unable to open $c: $!\n");
open (LOG, ">secedtem.log") or die ("unable to open log file: $!\n");
foreach (<LIST>)
{
    next unless ($_);
    $Command = $path."psexec.exe \\\$_ $o";
    @output = `"$Command"`; #backticks strike again
    print LOG "$Command\n";
    print LOG @output;
}
close (LOG);
close (LIST);

```