Microsoft DNS record updates using PowerShell and DNSCMD

UPDATE TO THIS POST:

Hi folks! some of the comments about records that are listed with (same as parent folder) as the name caused me to make an update. I've noted the updates in the following post:

http://www.discoposse.com/2013/04/14/updating-same-as-parent-folder-records-with-dnscmd-and-powershell/

I have also updated the code below to reflect the new code.

ORIGINAL POST:

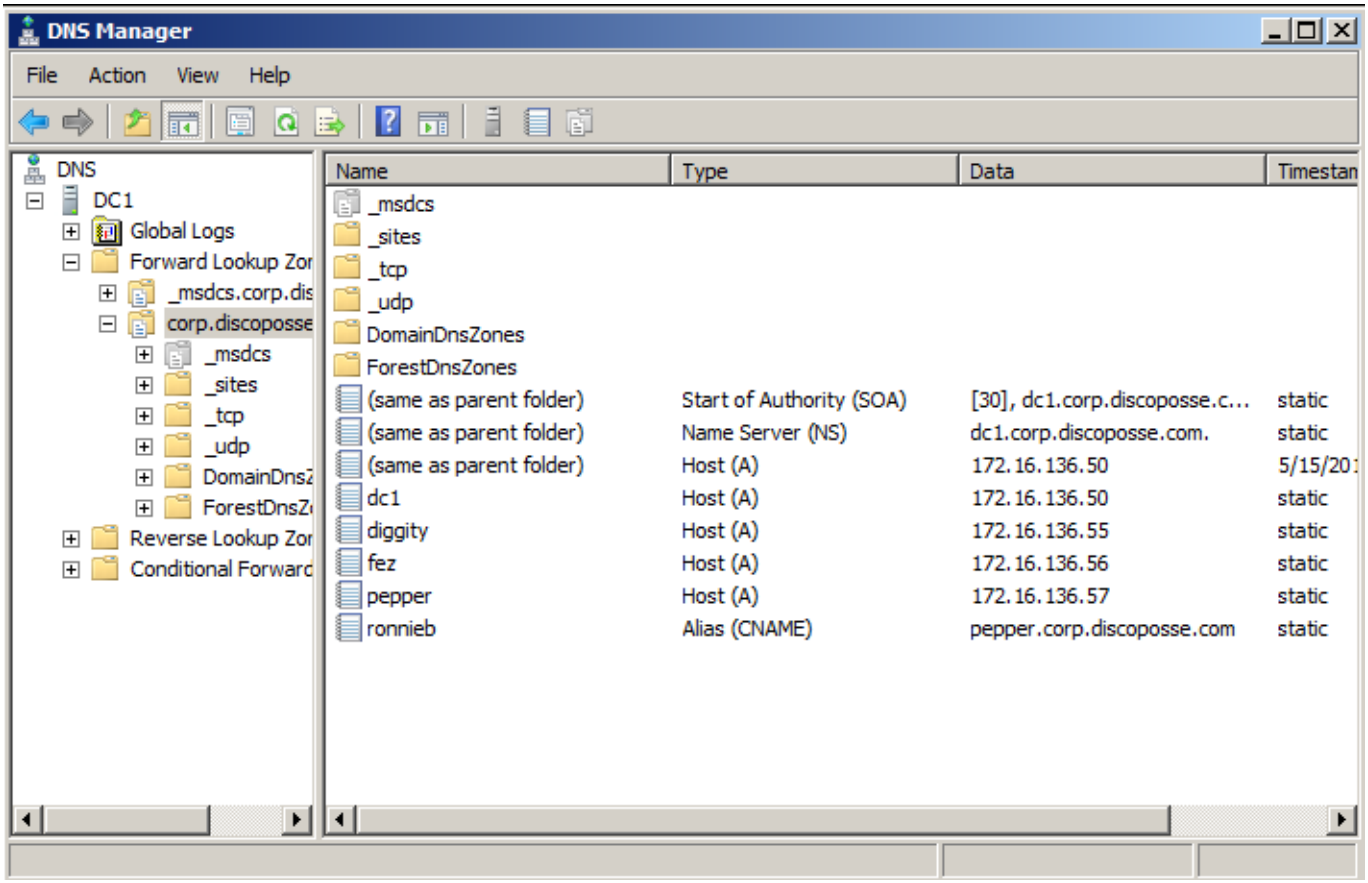
There are occasions where we need to do bulk DNS record management such as create and update a large series of records for IP network changes or BCP testing. The environment I am working with is a Microsoft DNS zone. It can be a standard Primary, or an Active Directory Integrated zone. Either type will work with the process we are creating here.

This is not a typical and common process as DNS is often self-managing and dynamic. Let's assume that you have a number of records from devices that cannot dynamically update, and you are currently using static A and CNAME records to define them.

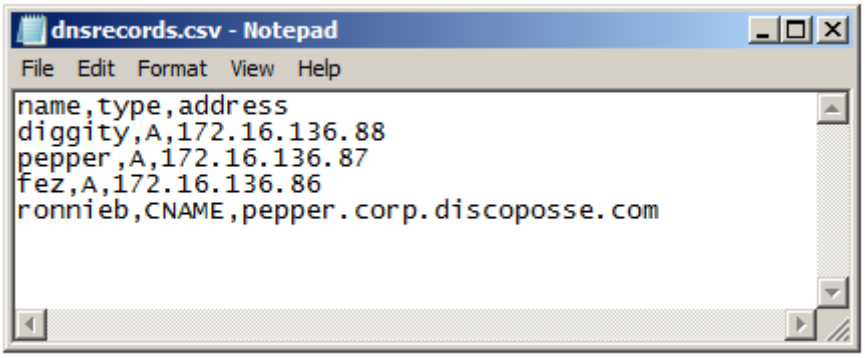
Because this is not a commonly occurring task, there are no native PowerShell CmdLets to do this. What we can do however is get the best of breed by using PowerShell to parse a file and craft the command line to use the Microsoft DNSCMD command line tool.

This process assumes that your records are all located in the root if the zone you define with the $DNSZone variable. The idea is that you can take this concept and flavour to taste for your particular needs.

The zone that we are managing for our example script is shown here:



To update the zone, we have to use a CSV file which is located in the same directory as the script, or you can define the full drive letter and path with the $InputFile variable if you so desire. The CSV file requires a header row which is **name,type,address** to give us our attributes for each object when we parse the file.



The script logic is fairly simple:

1. Define our zone info (server and zone name)
2. Import the CSV file into an array
3. Loop through the array
4. Create a delete command line and use the Invoke-Expression CmdLet to execute it
5. Create an add command line and use the Invoke-Expression CmdLet to execute it

I haven't added any fancy logging or anything, but there is some Write-Host CmdLets used to output the commands to screen so that you can ensure that they are coming out as expected.

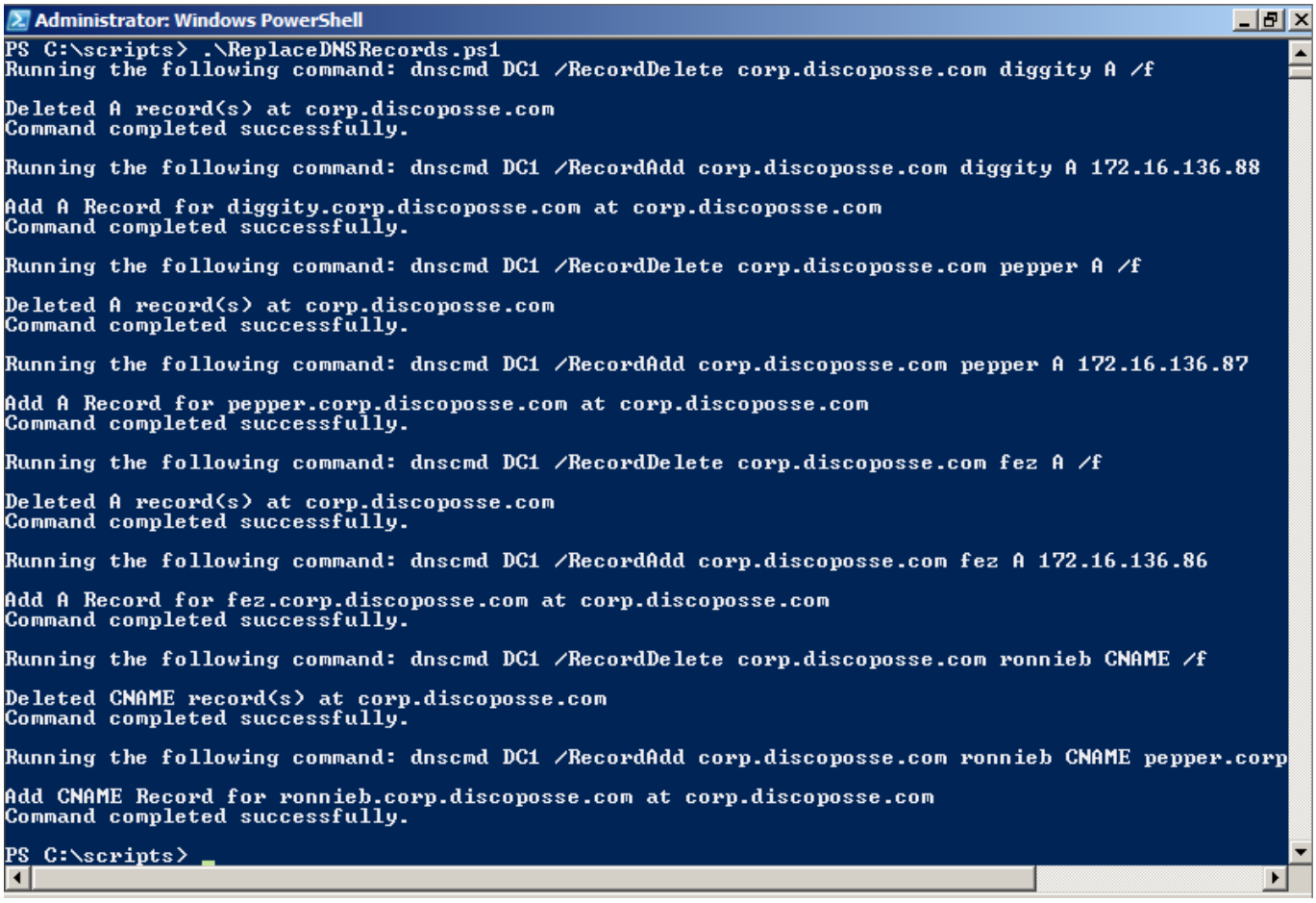
Fair warning on this one. Because we are using the Invoke-Expression CmdLet to launch the DNSCMD executable, we don't have the luxury of a -WhatIf option. You can comment out the Invoke-Expression lines and run using just the Write-Host output as a sanity check on your structure. And by "can" I mean you very definitely should!

The script can be downloaded at my TechNet Gallery
here: http://gallery.technet.microsoft.com/Update-DNS-records-with-da10910d

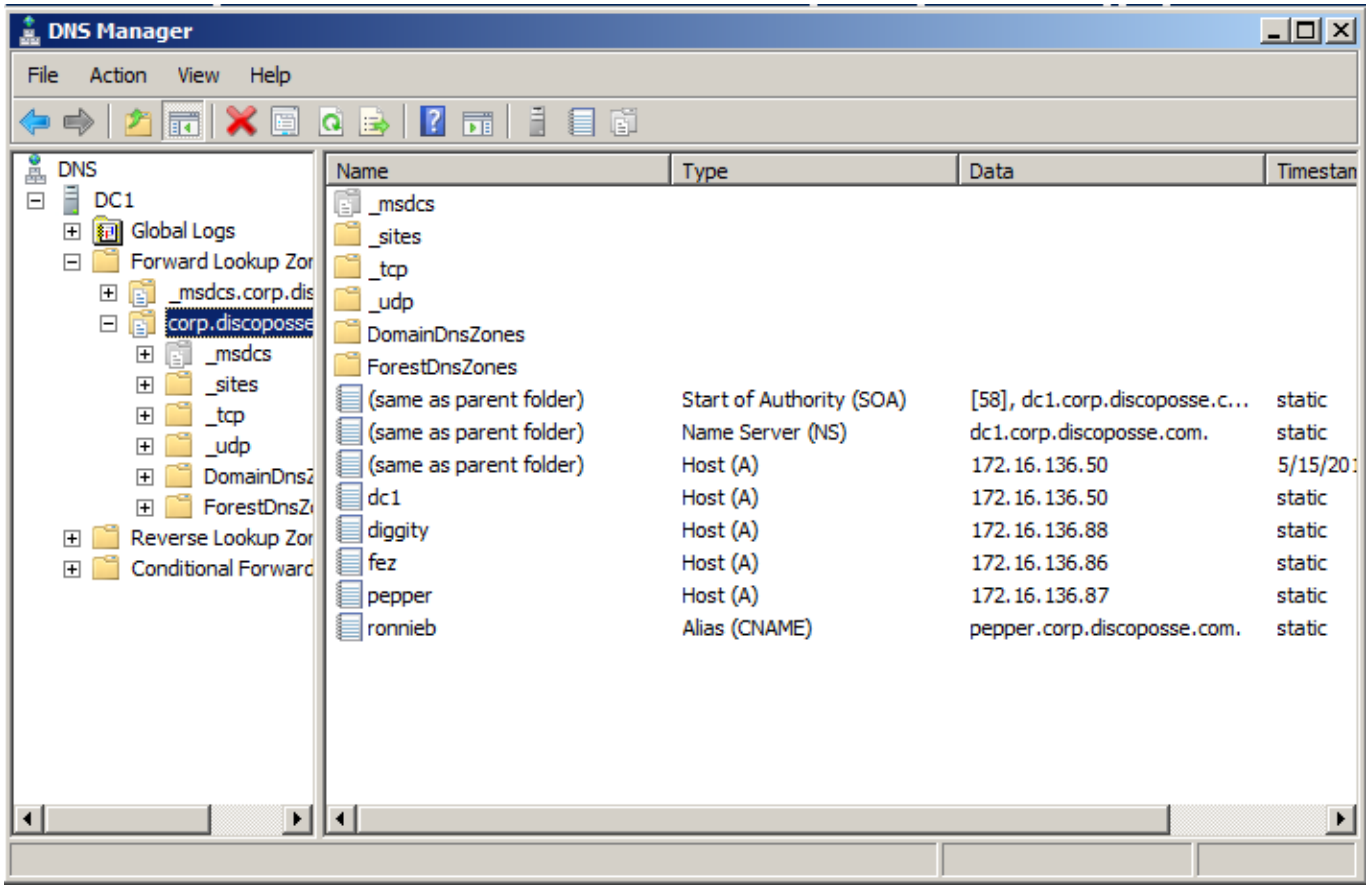
Here is the script:

```
# Environment Setup
$DNSServer = "DC1"
$DNSZone = "corp.discoposse.com"
$InputFile = "dnsrecords.csv"

# Read the input file which is formatted as name,type,address with a header row
$records = Import-CSV $InputFile

# Now we loop through the file to delete and re-create records
# DNSCMD does not have a modify option so we must use /RecordDelete first followed
by a /RecordAdd

ForEach ($record in $records) {

# Capture the record contents as variables
$recordName = $record.name
$recordType = $record.type
$recordAddress = $record.address

# Build our DNSCMD DELETE command syntax
$cmdDelete = "dnscmd $DNSServer /RecordDelete $DNSZone $recordName
$recordType $recordAddress /f"

# Build our DNSCMD ADD command syntax
$cmdAdd = "dnscmd $DNSServer /RecordAdd $DNSZone $recordName $recordType
$recordAddress"

# Now we execute the command
Write-Host "Running the following command: $cmdDelete"
Invoke-Expression $cmdDelete

Write-Host "Running the following command: $cmdAdd"
Invoke-Expression $cmdAdd
}
```

So let's run the script and see the results. This is the output from the script:



And now the resulting zone configuration shows the new updated records as per our CSV file:



While this may be a very simple example, it is meant to be a starting point to show you how to make the best of a situation where there is not a native PowerShell CmdLet. Being able to use hybrid scripting tools and techniques is a great thing to have in your toolkit of skills.

I hope that you find this useful!