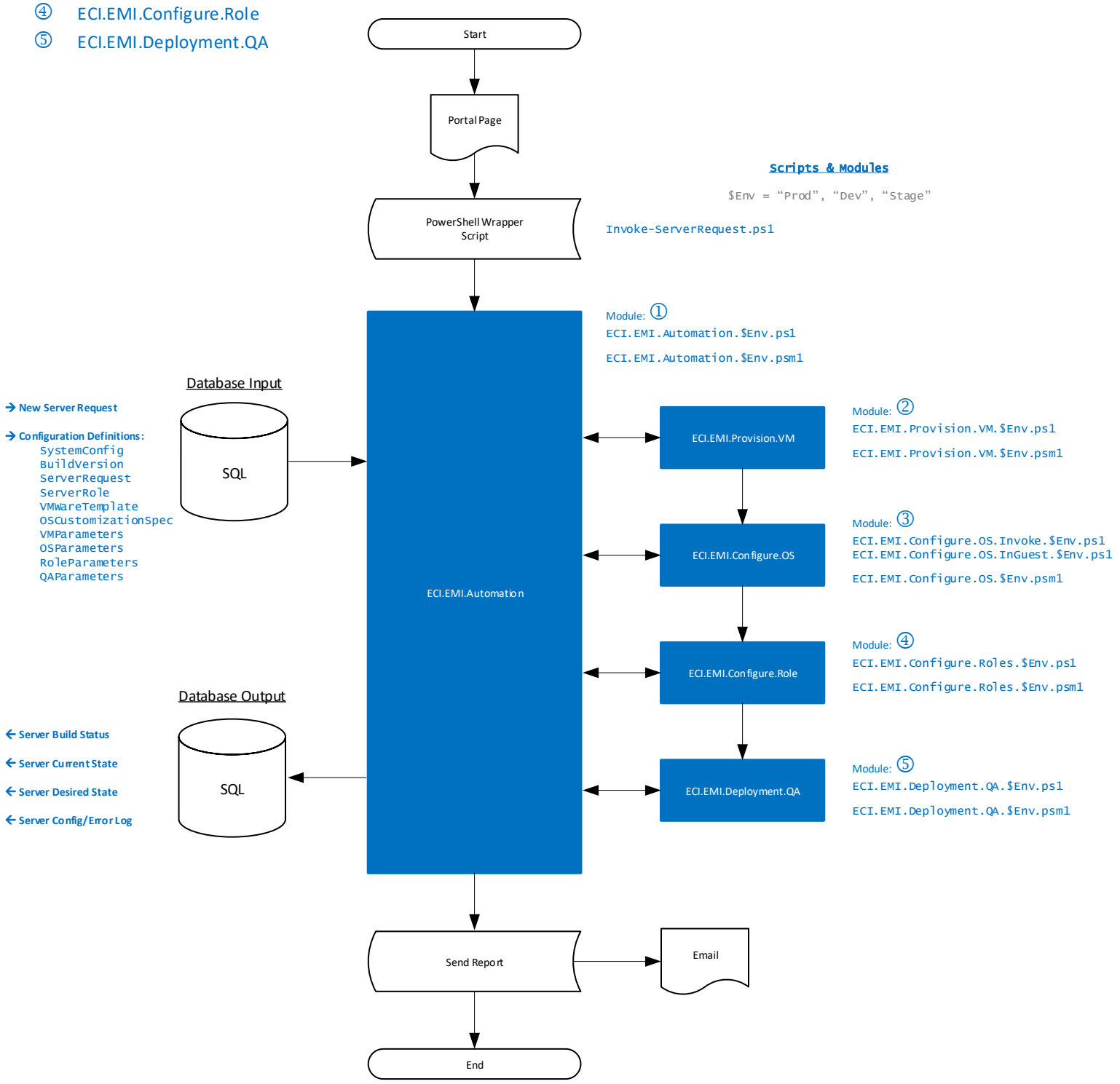
ECI EMI Server Provisioning Workflow

Process Overview

Modules:

① ECI.EMI.Automation
② ECI.EMI.Provison.VM
③ ECI.EMI.Configure.OS
④ ECI.EMI.Configure.Role
⑤ ECI.EMI.Deployment.QA



Chris Brennan
1/1/2019

ECI EMI Server Provisioning Workflow

Automation Process

Module: ①
ECI.EMI.Automation.$Env.ps1
ECI.EMI.Automation.$Env.psm1

→ **New Server Request**

→ **Configuration Definitions:**
- SystemConfig
- BuildVersion
- ServerRequest
- ServerRole
- VMWareTemplate
- OSCustomizationSpec
- VMParameters
- OSParameters
- RoleParameters
- QAParameters


```
### Create Transcript File Name
if($HostName)
{
    $global:TranscriptFile = $TranscriptPath +
"PowerShell_transcript" + "." + $HostName + "." + $Rnd + "." +
$TimeStamp + ".txt"
}
else
{
    $global:TranscriptFile = $TranscriptPath +
"PowerShell_transcript" + "." + $Rnd + "." + $TimeStamp + ".txt"
}
### Start Transcript Log
Start-Transcript -Path $TranscriptFile -NoClobber
```

```
### Set Default Configuration Mode
###--------------------------------------
if(!$ConfigurationMode)
{
    #$global:ConfigurationMode     = "Configure"
    $global:ConfigurationMode     = "Report"
}

elseif($ConfigurationMode)
{
    $global:ConfigurationMode = $ConfigurationMode
}
```

```
Importing Module: ECI.EMI.Automation.Prod
Importing Module: ECI.EMI.Automation.QA.Prod
Importing Module: ECI.EMI.Automation.VM.Invoke.Prod
Importing Module: ECI.EMI.Automation.VM.Prod
Importing Module: ECI.EMI.Automation.VM.Resources.Prod
Importing Module: ECI.EMI.Configure.OS.Prod
Importing Module: ECI.EMI.VM.Management.Prod
Importing Module: ECI.Functions
```

```
Get-ECI.EMI.Automation.SystemConfig -Env $Env
Get-ECI.EMI.Automation.BuildVersion
Get-ECI.EMI.Automation.ServerRequest
Get-ECI.EMI.Automation.ServerRole
Get-ECI.EMI.Automation.VMWareTemplate
Get-ECI.EMI.Automation.OSCustomizationSpec
Get-ECI.EMI.Automation.VMParameters
Get-ECI.EMI.Automation.OSParameters
```

```
Create-ECI.EMI.Automation.VMName -GPID $GPID -HostName $HostName

$global:VMName = $GPID + "_" + $HostName
```

```
$ServerStatus = @{
    RequestID    = $RequestID
    HostName     = $HostName
    ServerStatus = "Server Request Recieved"
}
Create-ECI.EMI.Automation.ServerStatus @ServerStatus
```

```
### Get/Set Server Record
###----------------------------
Check-ECI.EMI.Automation.ServerRecord -GPID $GPID -CWID $CWID -
HostName $HostName

if($ServerExists -eq $False)
{
    $ServerParams = @{
        RequestID      = $RequestID
        GPID           = $GPID
        CWID           = $CWID
        HostName       = $HostName
        ServerRole     = $ServerRole
        BuildVersion   = $BuildVersion
    }
    Create-ECI.EMI.Automation.ServerRecord @ServerParams
    Get-ECI.EMI.Automation.ServerID -RequestID $RequestID
}
elseif($ServerExists -eq $True)
{
    $global:ServerID = $ServerID
}
```

```
Create-ECI.EMI.Automation.CurrentStateRecord -ServerID (Get-
ECI.EMI.Automation.ServerID -RequestID $RequestID)
```

Provision VM

```
$StatusParams = @{
    RequestID          = $RequestID
    ServerID           = $ServerID
    HostName           = $HostName
    VerifyErrorCount   = 0
    Abort              = $False
    ServerStatus       = "VM Provisioning-Started"

}
Update-ECI.EMI.Automation.ServerStatus @StatusParams
```

```
$VMParams = @{
    ServerID           = $ServerID
    HostName           = $HostName
    ConfigurationMode  = $ConfigurationMode
    Env                = $Env
    Environment        = $Environment
}
Invoke-ECI.EMI.Provision.VM @VMParams
```

```
$ServerParams = @{
    ServerID           = $ServerID
    VMName             = $VMName
    ServerUUID         = $ServerUUID
    vCenterUUID        = $vCenterUUID
    vMoRef             = $vMoRef
}
Update-ECI.EMI.Automation.ServerRecord @ServerParams
```

```
$StatusParams = @{
    RequestID          = $RequestID
    ServerID           = $ServerID
    HostName           = $HostName
    VerifyErrorCount   = $VerifyErrorCount
    Abort              = $False
    ElapsedTime        = $VMElapsedTime
    ServerStatus       = "VM Provisioning-Completed"
}
Update-ECI.EMI.Automation.ServerStatus @StatusParams
```

Configure OS

```
$StatusParams = @{
    RequestID          = $RequestID
    ServerID           = $ServerID
    HostName           = $HostName
    VerifyErrorCount   = 0
    Abort              = $False
    ServerStatus       = "OS Configuration-Started"
}
Update-ECI.EMI.Automation.ServerStatus @StatusParams
```

```
$ConfigureOSParams = @{
    ServerID           = $ServerID
    HostName           = $HostName
    ConfigurationMode  = $ConfigurationMode
    Env                = $Env
    Environment        = $Environment
}
Invoke-ECI.EMI.Automation.OS @ConfigureOSParams
```

```
$StatusParams = @{
    RequestID          = $RequestID
    ServerID           = $ServerID
    HostName           = $HostName
    VerifyErrorCount   = $VerifyErrorCount
    Abort              = $False
    ElapsedTime        = $OSElapsedTime
    ServerStatus       = "OS Configuration-Completed"
}
Update-ECI.EMI.Automation.ServerStatus @StatusParams
```

```
$StatusParams = @{
    RequestID          = $RequestID
    ServerID           = $ServerID
    HostName           = $HostName
    VerifyErrorCount   = $VerifyErrorCount
    Abort              = $False
    ElapsedTime        = $OSElapsedTime
    ServerStatus       = "OS Configuration-Completed"
}
Update-ECI.EMI.Automation.ServerStatus @StatusParams
```

Configure Role

```
Invoke-ECI.EMI.Configure Role @RoleParams
```

Auto QA

```
Invoke-ECI.EMI.Auto.QA @QAParams
```

Completed

```
$StatusParams = @{
    RequestID          = $RequestID
    ServerID           = $ServerID
    HostName           = $HostName
    VerifyErrorCount   = $VerifyErrorCount
    Abort              = $False
    ElapsedTime        = $AutomationTime
    ServerStatus       = "Build Complete"
}
Update-ECI.EMI.Automation.ServerStatus @StatusParams
```

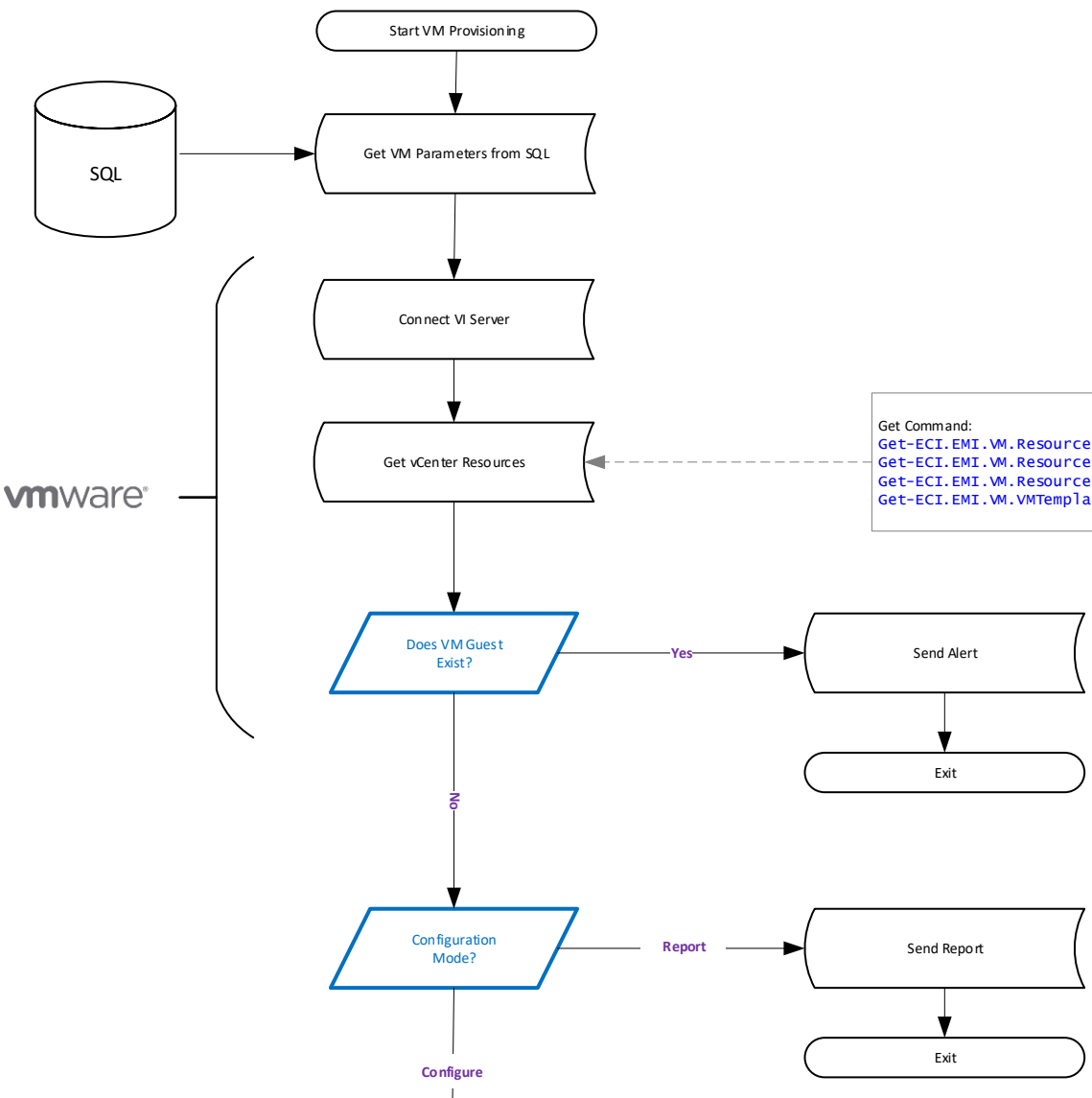
```
Get-ECI.EMI.Automation.ServerRequest-SQL      -RequestID $RequestID
Get-ECI.EMI.Automation.ServerRecord-SQL       -ServerID $ServerID
Get-ECI.EMI.Automation.ServerCurrentState-SQL -ServerID $ServerID
Get-ECI.EMI.Automation.ServerDesiredState-SQL -ServerID $ServerID
Get-ECI.EMI.Automation.ServerConfigLog-SQL    -ServerID $ServerID
```

ECI EMI Server Provisioning Workflow

VM Provisioning

Module: ②

ECI.EMI.Provision.VM.$Env.ps1
ECI.EMI.Provision.VM.$Env.psm1



Get Command:

```
Get-ECI.EMI.VM.Resources.ResourcePool -GPID $GPID
Get-ECI.EMI.VM.Resources.PortGroup -GPID $GPID
Get-ECI.EMI.VM.Resources.DataStore -ServerRole $ServerRole
Get-ECI.EMI.VM.VMTemplate -ServerRole $ServerRole
```

ECI CMDLET:
New-ECI.EMI.Automation.VM

New-ECI.EMI.Automation.VM

New VM Parameters:

```
### New VM Parameters
###-----------------------
$NewVMParameters = @{
    ConfigurationMode        = $ConfigurationMode
    VMName                   = $VMName
    ECIVMTemplate            = $ECIVMTemplate
    OSCustomizationSpecName  = $OSCustomizationSpecName
    ResourcePool             = $ResourcePool
    OSDataStore              = $OSDataStore
    PortGroup                = $PortGroup
    IPv4Address              = $IPv4Address
    SubnetMask               = $SubnetMask
    DefaultGateway           = $DefaultGateway
    PrimaryDNS               = $PrimaryDNS
    SecondaryDNS             = $SecondaryDNS
}
```

Create
OS Customization Spec

Set Command:

```
### New OSCustomizationSpec
###-----------------------
$OSCustomizationSpec = @{
    Name              = $OSCustomizationSpecName
    Type              = $Type
    OSType            = $OSType
    NamingScheme      = $NamingScheme.trim()
    FullName          = $FullName
    OrgName           = $OrgName
    AdminPassword     = $AdminPassword
    ChangeSid         = $True
    DeleteAccounts    = $False
    TimeZone          = $TimeZone
    ProductKey        = $ProductKey
    LicenseMode       = $LicenseMode
    Workgroup         = $Workgroup
}
New-OSCustomizationSpec @OSCustomizationSpec
```

Create
Customization NIC Mapping

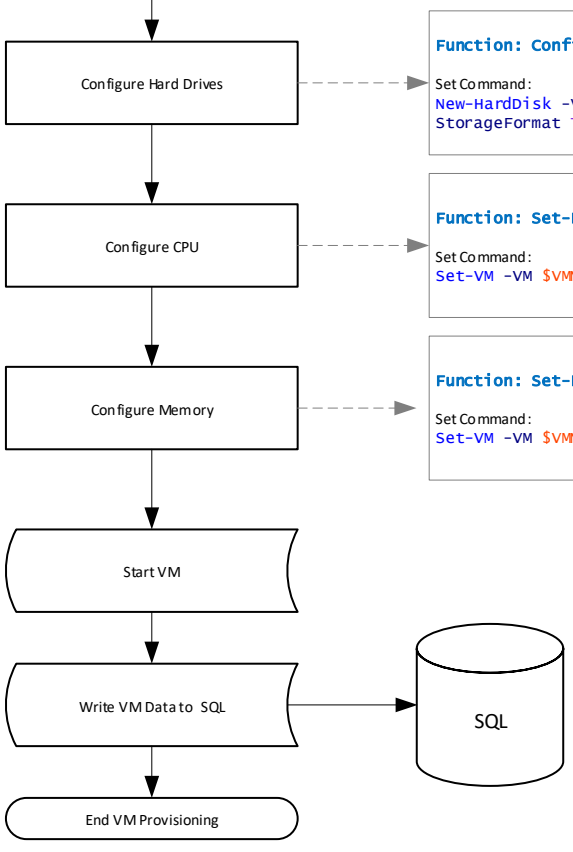
Set Command:

```
### NIC Mapping
###-----------------------
$NicMapping = @{
IPMode          = "UseStaticIp "
IpAddress       = $IPv4Address
SubnetMask      = $SubnetMask
DefaultGateway  = $DefaultGateway
Dns             = $PrimaryDNS,$SecondaryDNS
}

Get-OSCustomizationSpec $OSCustomizationSpecName |
Get-OSCustomizationNicMapping |
Set-OSCustomizationNicMapping @NICMapping
```

Create
New VM

Set Command:

```
### New VM
$NewVMParams = {
    Name                = $VMName
    Template            = $ECIVMTemplate
    Location            = $vCenterFolder
    ResourcePool        = $ResourcePool
    Datastore           = $OSDataStore
    OSCustomizationSpec = $OSCustomizationSpecName
}
New-VM @NewVMParams
```



Function: Configure-ECI.EMI.Automation.VM.HardDisks

Set Command:

```
New-HardDisk -VM $VMName -Datastore $DataStore -CapacityGB $CapacityGB -
StorageFormat Thin -Persistence persistent -Confirm:$false
```

Function: Set-ECI.EMI.Automation.VM.VMCPU

Set Command:

```
Set-VM -VM $VMName -Confirm:$False -NumCpu $vCPUCount
```

Function: Set-ECI.EMI.Automation.VM.VMMemory

Set Command:

```
Set-VM -VM $VMName -Confirm:$False -MemoryGB $vMemoryGB
```

OS Configuration

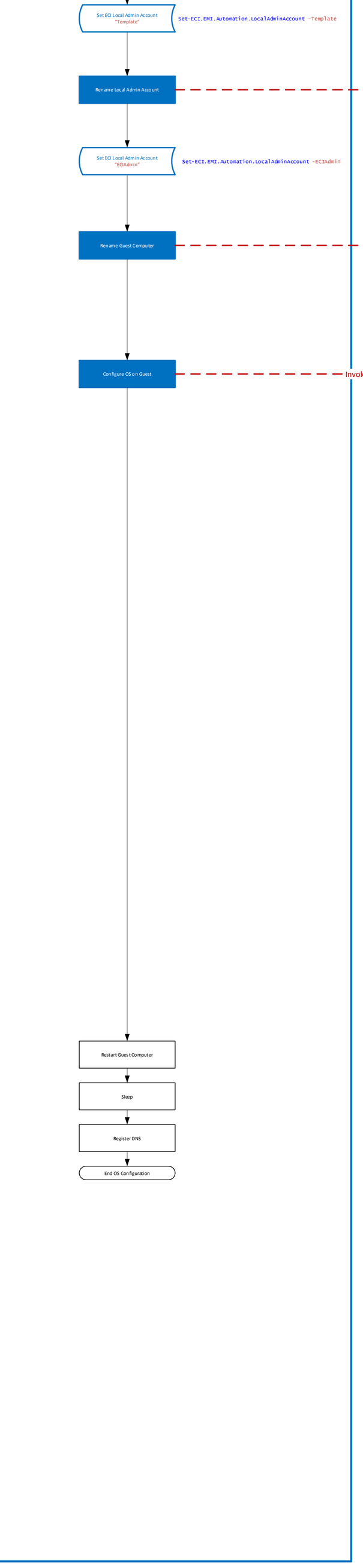
Module: ③

`ECI.EMI.Configure.OS.Invoke.$Env.ps1`
`ECI.EMI.Configure.OS.InGuest.$Env.ps1`

`ECI.EMI.Configure.OS.$Env.psm1`

Deployment Server

`ECI.EMI.Configure.OS.Invoke.$Env.ps1`



In-Guest OS

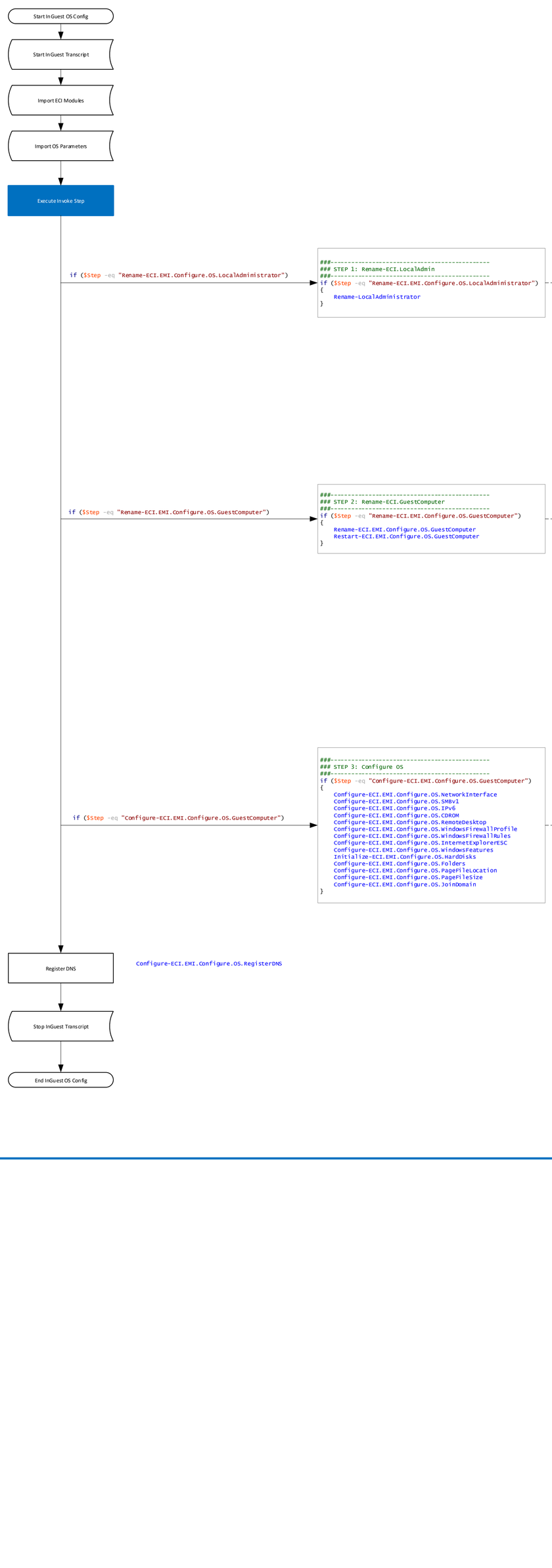
```
$ExecutionPolicyonVMGuest =
{
    if ($(Get-ExecutionPolicy) -ne "Bypass")
    {
        Write-Host "`nSetting Execution Policy on VM Guest to Bypass:"
        Set-ExecutionPolicy ByPass -Scope LocalMachine
    }
    else
    {
        Write-Host "`nExecution Policy on VM Guest already set to Bypass"
    }
}
$Invoke = Invoke-VMScript -VM $VMName -ScriptText $ExecutionPolicyonVMGuest -ScriptType Powershell -GuestUser $Creds.LocalAdminName -GuestPassword
$Creds.LocalAdminPassword
```

```
SecondaryDNS,10.126.1.2
IPv6Preference,True
OSParameterID,1
IPv4Address,10.126.1.11
RemoteDesktopPreference,True
DomainUserName,cbrennan@eci.com
GPID,ETEST040
CDROMLetter,R
ClientDomain,ercolanomgmt.corp
CWID,0
RDPRestrictionsPreference,False
InternetExplorerESCPreference,False
PrimaryDNS,10.126.1.1
ECILocalAdminPassword,cH3rOk33
BuildVersion,1.0.0
ECILocalAdminName,ECIAdmin
ServerRole,2016Server
WindowsFirewallPreference,True
BackupRecovery,True
PageFileLocation,D
SMBv1,False
InstanceLocation,LD5
RequestServerRole,2016Server
DisasterRecovery,False
PageFileMultiplier,1.2
NetworkInterfaceName,Ethernet1
HostName,Test-LD5-Oct07
SubnetMask,255.255.255.224
RequestDateTime,11/18/2018 08:46:19
DefaultGateway,10.126.1.30
RequestID,1073
AdministrativeUserName,cbrennan@ercolanomgmt.corp
AutomationLogPath,C:\Scripts\_VMAutomationLogs
ConfigurationMode,Configure
AdministrativePassword,Tolkien4374
```

```
Importing Module: ECI.EMI.Automation.Prod
Importing Module: ECI.EMI.Automation.QA.Prod
Importing Module: ECI.EMI.Automation.VM.Invoke.Prod
Importing Module: ECI.EMI.Automation.VM.Prod
Importing Module: ECI.EMI.Automation.VM.Resources.Prod
Importing Module: ECI.EMI.Configure.OS.Prod
Importing Module: ECI.EMI.VM.Management.Prod
Importing Module: ECI.Functions
```

In-Guest Script:

`ECI.EMI.Configure.OS.InGuest.$Env.ps1`




```
###-------------------------------------------
### STEP 1: Rename-ECI.LocalAdmin
###-------------------------------------------
if ($Step -eq "Rename-ECI.EMI.Configure.OS.LocalAdministrator")
{
    Rename-LocalAdministrator
}
```

```
###-------------------------------------------
### STEP 2: Rename-ECI.GuestComputer
###-------------------------------------------
if ($Step -eq "Rename-ECI.EMI.Configure.OS.GuestComputer")
{
    Rename-ECI.EMI.Configure.OS.GuestComputer
    Restart-ECI.EMI.Configure.OS.GuestComputer
}
```

```
###-------------------------------------------
### STEP 3: Configure OS
###-------------------------------------------
if ($Step -eq "Configure-ECI.EMI.Configure.OS.GuestComputer")
{
    Configure-ECI.EMI.Configure.OS.NetworkInterface
    Configure-ECI.EMI.Configure.OS.SMBv1
    Configure-ECI.EMI.Configure.OS.IPv6
    Configure-ECI.EMI.Configure.OS.CDROM
    Configure-ECI.EMI.Configure.OS.RemoteDesktop
    Configure-ECI.EMI.Configure.OS.WindowsFirewallProfile
    Configure-ECI.EMI.Configure.OS.WindowsFirewallRules
    Configure-ECI.EMI.Configure.OS.InternetExplorerESC
    Configure-ECI.EMI.Configure.OS.WindowsFeatures
    Initialize-ECI.EMI.Configure.OS.HardDisks
    Configure-ECI.EMI.Configure.OS.Folders
    Configure-ECI.EMI.Configure.OS.PageFileLocation
    Configure-ECI.EMI.Configure.OS.PageFileSize
    Configure-ECI.EMI.Configure.OS.JoinDomain
}
```

Module:

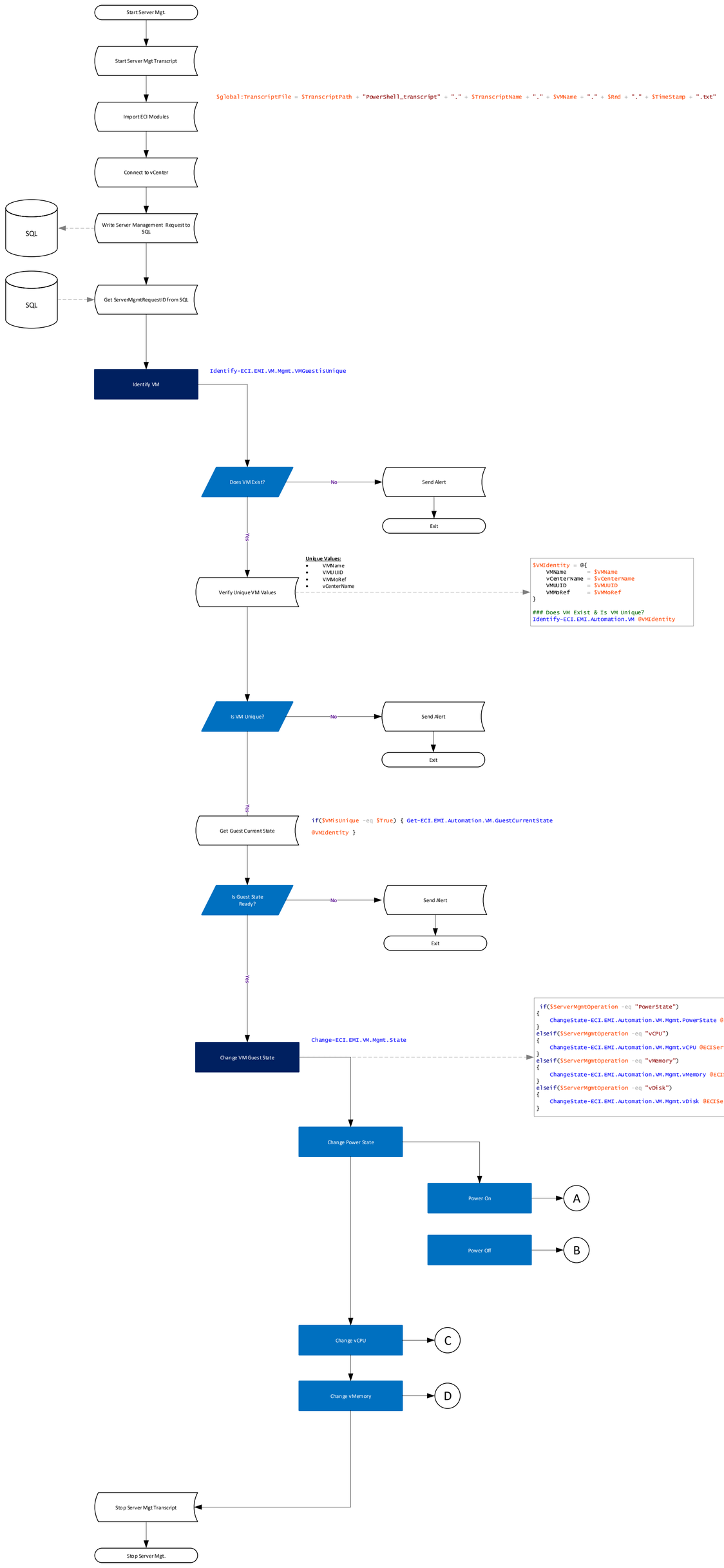
`ECI.EMI.Configure.OS.InGuest.$Env.psm1`

```
$RenameLocalAdmin ={
    ### Use .NET to Find the Current Local Administrator Account
    Add-Type -AssemblyName System.DirectoryServices.AccountManagement
    $ComputerName       = [System.Net.Dns]::GetHostName()
    $PrincipalContext   = New-Object
System.DirectoryServices.AccountManagement.PrincipalContext([System.DirectoryServices.AccountManagement.ContextType]::Machine,
$ComputerName)
    $UserPrincipal      = New-Object System.DirectoryServices.AccountManagement.UserPrincipal($PrincipalContext)
    $Searcher           = New-Object System.DirectoryServices.AccountManagement.PrincipalSearcher
    $Searcher.QueryFilter = $UserPrincipal

    ### The Administrator account is the only account that has a SID that ends with "-500"
    $Account = $Searcher.FindAll() | Where-Object {$_.Sid -Like "*-500"}
    $CurrentAdminName = $Account.Name

    $ECILocalAdminName = "#ECILocalAdminName#"

    Rename-LocalUser -Name $CurrentAdminName -NewName $ECILocalAdminName -ErrorAction SilentlyContinue | Out-Null
}
```

```
########################################
### GET CURRENT CONFIGURATION STATE:
########################################
[scriptblock]$Script:GetCurrentState =
{
    $global:CurrentState =    (Get-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName) ComputerName
}

########################################
### SET DESIRED-STATE:
########################################
[scriptblock]$Script:SetDesiredState =
{
    Rename-Computer -ComputerName . -NewName $HostName
}
```

ECI EMI Server Provisioning Workflow

Server Management



Server Management Process:

- Does the VM Exist?
- Verify the VM is Unique? (VMName, vCenterName, VMUUID, VMMoRef)
- Is Guest State Ready?
- Execute VM State Change Function
  - PowerOn
  - PowerOff
  - Change vCPU
  - Change vMemory
- Verify VM State Change
- Report VM State Change to SQL

```
function Identify-ECI.EMI.Automation.VM
{
    Param(
        [Parameter(Mandatory = $True)][string]$VMName,
        [Parameter(Mandatory = $True)][string]$vCenterName,
        [Parameter(Mandatory = $True)][string]$VMUUID,
        [Parameter(Mandatory = $True)][string]$VMMoRefID
    )

    $VM = Get-VM -Name $VMName -ErrorAction SilentlyContinue

    if($VM)
    {
        $VMExists = $True

        ### Get Unique VM Values
        ### ------------------------------------
        $verifyVMName      = $VM.Name
        $verifyVMUUID      = $VM | %{(Get-View $_.Id).config.uuid}
        $verifyVMMoRefID   = $VM.ID
        $verifyvCenterName = (($VM.UID).split("@")[1]).split(":")[0]

        ### Identify Unique VM
        ### ------------------------------------
        if(($VMName -eq $verifyVMName) -AND ($vCenterName -eq $verifyvCenterName) -AND ($VMUUID -eq $verifyVMUUID) -AND ($VMMoRefID -eq $verifyVMMoRefID))
        {
            $VMUnique = $True
        }
        else
        {
            $VMUnique = $False
        }
    }
    elseif(!$VM)
    {
        $VMExists = $False
    }
    Return $VMUnique
}
```

ECI EMI Server Provisioning Workflow

Change Power State – Power Off

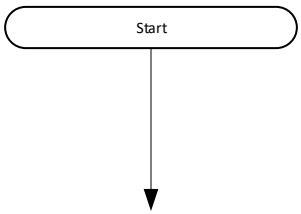
Report-ECI.EMI.VM.GuestStateChange

Role Configuration

Module: ④

`ECI.EMI.Configure.Role.$Env.ps1`

`ECI.EMI.Configure.Role.$Env.psm1`

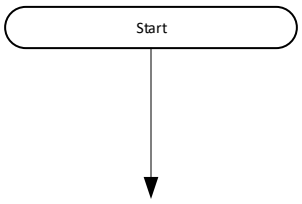


Auto QA

Module: ⑤

`ECI.EMI.Auto.QA.$Env.ps1`

`ECI.EMI.Auto.QA.$Env.psm1`



Portal Wrapper

Module: A.

`Invoke-ServerRequest.ps1`