# Server not communicating with our Data Center (Agent Malfunction)

**Logic: Agent Malfunction**

If agent is not reporting its live status to DC, but LMI is reporting its availability every 5 minutes than, system will wait up to 80 minutes, and later will trigger an Agent-Malfunction alert.

* **Manual Steps:**

1. Check Agent type, if DPMA exit the script and if MSMA continue.
2. Check Agent services [If stopped/Disable we will start the service and if found any error while starting we will exit the script].

Also, if service is in start state we will restart the service and if found any error while re-starting we will exit the script

1. Check RMM Agent version, it should be latest. [If not we will update the Agent]
2. Check SAAZOD directory files, it should be 180 plus and the extensions (Extension are common if the files are encrypted).
3. Check Agent-Installedpath/SaazMobile.ini(Should be of 1KB or more) file and if the machine has proxy then is the proxy details updated on the SaazMobile.ini
4. Check if server has Proxy (If yes check does SaazMobile.ini has the entry of Proxy)
5. Check our webposting link is accessible via Internet Explorer or not with certificate issued by valid DegiCert
6. Verify the Agent communication logs, which will have the details.
7. We will pass parameter if checked yes, we will restart the Rmm Agent service once.

* **Automation Logic:**

To design an automation workflow for scenario where Server is not communicating with our DataCenter. We will only consider the following flow:

1. We will fetch all the above details.
2. If needed perform the remedy steps from the script it self.

**Below are the PowerShell Command that can used to create the script.**

**Task 1**

1. **Verify the system’s Windows NT kernel version.**

**(Get-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion").CurrentVersion**

**C:\Users\sagar.kukreti\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\7B289488.tmp**

|  |  |
| --- | --- |
| **File Version** | **OS** |
| **5** | **Windows 2000** |
| **5.2** | **Windows Server 2003** |
| **6** | **Windows Server 2008** |
|  |  |
| **File Version** | **OS** |
| **6.1** | **Windows Server 2008 R2** |
| **6.2** | **Windows Server 2012** |
| **6.3** | **Windows Server 2012 R2** |
| **10** | **Windows Server 2016** |
| **10** | **Windows Server 2019** |
| **10** | **Windows Server 2019** |

**If the kernel version is less than 6.1, exit with print a message: Legacy OS found.**

**If the kernel version is equal to OR greater than 6.1, continue to next step.**

1. **Get the OS version:**

**(Get-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion").ProductName**



**Task 1**

* **Check if Server/machine has DPMA or MSMA Agent**
* **If DPMA, exit the code.**

*[String]$productype = (Get-ItemProperty -Path "Registry::HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\SAAZOD\" -erroraction silentlycontinue).Type*

*[String]$productype1 = (Get-ItemProperty -Path "Registry::HKEY\_LOCAL\_MACHINE\SOFTWARE\SAAZOD\" -erroraction silentlycontinue).Type*

*If ($productype -eq "DPMA" -or $productype1 -eq "DPMA") {*

*Write-Output "[MSG]: Machine has a DPMA Agent installed. Kindly run the script on Server Agent."*

*Exit;*

*}*

**Task 2 :**

**Below code will be used in multiple function.**

* **For getting the Agent installation path.**

*if ($env:PROCESSOR\_ARCHITECTURE -eq "AMD64") {*

*$Path = (Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\SAAZOD")*

*}*

*elseif ($env:PROCESSOR\_ARCHITECTURE -eq "x86") {*

*$Path = (Get-ItemProperty "HKLM:\SOFTWARE\SAAZOD"*

*}*

**Task 3**

1. **Check Agent version**
2. **Check Agent install Path**
3. **Below files details and if any of the below file is missing we will exit the code.**

* ***SaazMobile.ini : "$($Path.InstallationPath)\Saazmobile.ini (With the size)***
* ***SaazServerPlus.ini : "$($Path.InstallationPath)\SaazServerPlus.ini***
* ***SaazScheduler.ini: "$($Path.InstallationPath)\Configuration\SaazScheduler.ini" (With the size)***

**32-bit** : "HKLM:\SOFTWARE \SAAZOD" – ValueName = InstallationPath

**64-bit** : "HKLM:\SOFTWARE\Wow6432Node\SAAZOD" – ValueName = InstallationPath

*if ($env:PROCESSOR\_ARCHITECTURE -eq "AMD64") {*

*$Version = (Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\SAAZOD").DisplayVersion*

*Write-Output "RMM Agent Version : $Version"*

*}*

*elseif ($env:PROCESSOR\_ARCHITECTURE -eq "x86") {*

*$Version = (Get-ItemProperty "HKLM:\SOFTWARE\SAAZOD").DisplayVersion*

*Write-Output "RMM Agent Version : $Version"*

*}*

*function Get-RMMDetails {*

*try {*

*#changes*

*if ($env:PROCESSOR\_ARCHITECTURE -eq "AMD64") {*

*$Path = (Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\SAAZOD")*

*}*

*elseif ($env:PROCESSOR\_ARCHITECTURE -eq "x86") {*

*$Path = (Get-ItemProperty "HKLM:\SOFTWARE\SAAZOD")*

*}*

*$Files = Get-ChildItem $Path.InstallationPath*

*if (Test-Path "$($Path.InstallationPath)\Saazmobile.ini") {*

*$x = "Present"*

*$Size = [Math]::Round(((Get-Item "$($Path.InstallationPath)\SAAZMobile.ini").length) / 1KB, 2)*

*}*

*else { $x = "Absent"; $Size = "NA" }*

*if (Test-Path "$($Path.InstallationPath)\SaazServerPlus.ini") { $y = "Present" }else { $y = "Absent" }*

*if (Test-Path "$($Path.InstallationPath)\Configuration\SaazScheduler.ini") {*

*$z = "Present"*

*$Size1 = [Math]::Round(((Get-Item "$($Path.InstallationPath)\Configuration\SaazScheduler.ini").length) / 1KB, 2)*

*}*

*else { $z = "Absent"; $Size = "NA" }*

*if ($x -eq "Absent" -or $y -eq "Absent" -or $z -eq "Absent") {*

*Write-Output @"*

*Rmm Agent Version : $($Path.DisplayVersion)*

*Install Path : $($Path.InstallationPath)*

*SaazMobile.ini : $x (Size : $Size)*

*SaazServerPlus.ini : $y*

*SaazScheduler.ini : $z (Size : $Size1)*

*Agent File count : $($Files.Count)*

*"@*

*}*

*else {*

*Write-Output @"*

*Rmm Agent Version : $($Path.DisplayVersion)*

*Install Path : $($Path.InstallationPath)*

*SaazMobile.ini : $x (Size : $Size KB)*

*SaazServerPlus.ini : $y*

*SaazScheduler.ini : $z (Size : $Size1 KB)*

*Agent File count : $($Files.Count)*

*"@*

*}*

*}*

*catch {*

*Write-Output "$($\_.Exception.message)"*

*}*

*}*

**Task 4**

* **Check Rmm Agent Services details.**
* **Services start mode should be Automatic and status should be Running .**
* **If services is in stop or disabled, take appropriate action on the service. (Provide one Variable, if set has True then changes should get applicable)**
* **Again will check the services and if found any service is not started we will print the details and exit from the script.**

**Note**: Using WMIObject as 2K8R2 does not have Startup type property.

Get-WmiObject Win32\_Service | Where {($\_.name -like "Saaz\*") -and ("SAAZRemoteSupport","SAAZRCCTL", "SAAZapsc" -notcontains $\_.Name)} | Select Name,State,StartMode | Format-List

* If services are in Stopped/Disabled state, we will print the details.
* Then, will start the services with Automatic state

**Below function has one Parameter which needs to be passed $true if we want to restart or start the services.**

Variable name : [Bool]$ServiceStart=$true

*function Get-RMMServices {*

*$ErrorActionPreference = "Stop"*

*Write-Output "`nBelow are the RMM agent services status:"*

*try {*

*$RmmService = Get-WmiObject Win32\_Service | Where-Object { ($\_.name -like "Saaz\*") -and ("SAAZRemoteSupport", "SAAZRCCTL", "SAAZapsc" -notcontains $\_.Name) } | Select-Object Name, State, StartMode*

*$RmmService | ForEach-Object {*

*write-output @"*

*`nName : $($\_.Name)*

*State : $($\_.State)*

*StartMode : $($\_.StartMode)*

*"@*

*}*

*if ($ServiceStart) {*

*$Startmodes = $RmmService | ForEach-Object { $\_.Startmode }*

*$Status = $RmmService | ForEach-Object { $\_.State }*

*$Servicestatus=@()*

*$Servicestatus1=@()*

*if ($Startmodes -contains "Disabled" -or "Manual") {*

*foreach ($service in $RmmService) {*

*try {*

*Set-Service -Name $service.name -StartupType Automatic -PassThru | Start-Service*

*#Write-Output "$($service.name) has been successfully started"*

*$Servicestatus += $service.name*

*}*

*catch {*

*#Write-Output "$($service.name) service failed to start"*

*$Servicestatus1 += $service.name*

*}*

*}*

*}*

*elseif ($Status -contains "Running" -or "Stopped") {*

*foreach ($service in $RmmService) {*

*try {*

*Restart-Service -Name $service.name*

*$Servicestatus += $service.name*

*}*

*catch {*

*$Servicestatus1 += $($service.name)*

*}*

*}*

*}*

*if ($Servicestatus) {*

*Write-output "`nBelow service has been successfully started/restarted:`n"*

*$Servicestatus*

*}*

*if ($Servicestatus1) {*

*Write-output "`nFailed to restart below services:`n"*

*$Servicestatus1*

*}*

*#changes*

*$RmmService1 = Get-WmiObject Win32\_Service | Where-Object { ($\_.name -like "Saaz\*") -and ("SAAZRemoteSupport", "SAAZRCCTL", "SAAZapsc" -notcontains $\_.Name) } | Select-Object Name, State, StartMode*

*$Status1 = $RmmService1 | Where-Object { $\_.State -eq "stopped" }*

*if ($status1) {*

*Write-Output "`nService still in stopped state"*

*$Status1 | ForEach-Object {*

*Write-output @"*

*`nName : $($\_.name)*

*State : $($\_.State)*

*StartMode : $($\_.StartMode)*

*"@*

*}*

*break;*

*}*

*}*

*}*

*catch {*

*Write-Output "$($\_.Exception.message)"*

*}*

*}*

**Task 5**

* **Check Proxy details on the server.**
* **If proxy available, then read the content in SaazMobile.ini file (Below lines we have to read).**

**[ProxySettings]**

**ProxyIP=**

**ProxyPort=**

**ProxyUserName=**

**ProxyPassword=**

*Function Get-ProxyDetails {*

*try {*

*$Conprx = (Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Connections" -Name WinHttpSettings).WinHttpSettings*

*}*

*catch {*

*$Conprx = $null*

*}*

*finally {*

*if ($Conprx) {*

*Write-Output "`nProxy details:"*

*$proxylength = $Conprx[12]*

*if ($proxylength -gt 0) {*

*$proxy = -join ($Conprx[(12 + 3 + 1)..(12 + 3 + 1 + $proxylength - 1)] | ForEach-Object { ([char]$\_) })*

*$bypasslength = $Conprx[(12 + 3 + 1 + $proxylength)]*

*if ($bypasslength -gt 0) {*

*$bypasslist = -join ($Conprx[(12 + 3 + 1 + $proxylength + 3 + 1)..(12 + 3 + 1 + $proxylength + 3 + 1 + $bypasslength)] | ForEach-Object { ([char]$\_) })*

*}*

*else {*

*$bypasslist = '(none)'*

*}*

*Write-Output @"*

*Winhttpproxy : $proxy*

*Winhttpproxybypasslist : $bypasslist*

*"@*

*$ProxyEnabled = $true*

*}*

*else {*

*Write-Output @"*

*Winhttpproxy : Direct Access*

*Winhttpproxybypasslist : (none)*

*"@*

*}*

*}*

*else {*

*Write-Output @"*

*Winhttpproxy : error - not able to read registry entry*

*Winhttpproxybypasslist : error - not able to read registry entry*

*"@*

*}*

*}*

*if ($ProxyEnabled) {*

*$x = Get-Content "C:\Program Files (x86)\SAAZOD\SAAZMobile.ini"*

*Write-Output "`nBelow is the SaazMobile.ini Proxy details `n"*

*$x | Where-Object { $\_ -like "Proxy\*" }*

*}*

*}*

**Task 6**

* **Check if our webposting site is accessible or not.**
* **If yes, check certificate issued to the site should be : DigiCert**
* **If certificate is not DigiCert exit**
* **If site is not accessible, we will perform below steps and exit.**
* **Ping “webpost.itsupport247.com”**
* **telnet Port 80 and 443 is listening “webpost.itsupport247.com”**
* **nslookup site “webpost.itsupport247.com” and “google.com”**

**Below function will capture details. (Note we would need to add Ping, ping and nslookup details)**

*Function Get-Webpostlink {*

*try {*

*$web = New-Object System.Net.WebClient*

*Write-Output "`nWebposting link output:"*

*$web.DownloadString("https://webpost.itsupport247.net/webpost.itsupport.asp") -split "<br>"*

*try {*

*$URL = "Webpost.itsupport247.net"*

*$Port = 443*

*$tcpsocket = New-Object Net.Sockets.TcpClient($URL, $Port)*

*$tcpstream = $tcpsocket.GetStream()*

*$sslStream = New-Object System.Net.Security.SslStream($tcpstream, $false)*

*$sslStream.AuthenticateAsClient($URL)*

*$CertInfo = New-Object system.security.cryptography.x509certificates.x509certificate2($sslStream.RemoteCertificate)*

*Write-Output @"*

*`nWeb-posting link Certificate details :*

*Issued By : $($CertInfo.Issuer)*

*Valid From : $($CertInfo.NotBefore)*

*Valid Till : $($CertInfo.NotAfter)*

*"@*

*}*

*catch {*

*Write-Output @"*

*`nCertificate details*

*Unable to get certificate details for `"$URL`".*

*"@*

*}*

*}*

*catch {*

*if ($\_.Exception -match "Unable to connect to the remote server") {*

*Write-Output "[Err] : Unable to connect to the remote server"*

*}*

*elseif ($\_.Exception -match "The remote name could not be resolved") {*

*Write-Output "[Err] : The remote name could not be resolved"*

*}*

*else {*

*Write-Ouput "$($\_.Exception.message)"*

*}*

*}*

*}*

**Task 7**

* **Execute agent binaries for communicating the server with our datacenter forcefully**

**ZWbPe.exe**

**DMPHelpDesk.exe /iamlive**

**DMPHelpDesk.exe /uploadstatus**

**ZWbPe.exe**

*Function Set-BinExecution {*

*try {*

*$ErrorActionPreference = "SilentlyContinue"*

*if ($env:PROCESSOR\_ARCHITECTURE -eq "AMD64") {*

*$Path = (Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\SAAZOD")*

*$EXE = "C:\\PROGRA~2\\SAAZOD"*

*}*

*elseif ($env:PROCESSOR\_ARCHITECTURE -eq "x86") {*

*$Path = (Get-ItemProperty "HKLM:\SOFTWARE\SAAZOD")*

*$EXE = "C:\\PROGRA~2\\SAAZOD"*

*}*

*if (-not [System.IO.File]::Exists("$($Path.InstallationPath)\ZWbPe.exe")) {*

*Write-Output "`n[Err] : ZWbPe.exe file does not exists"*

*Break;*

*}*

*if (-not [System.IO.File]::Exists("$($Path.InstallationPath)\DMPHelpDesk.exe")) {*

*Write-Output "`n[Err] : DMPHelpDesk file does not exists"*

*Break;*

*}*

*$Version = (Get-WmiObject -Class CIM\_DataFile -Filter "Name = '$EXE\\ZWbPe.exe'").version*

*$Version1 = (Get-WmiObject -Class CIM\_DataFile -Filter "Name = '$EXE\\DMPHelpDesk.exe'").version*

*if ($Version -and $Version1) {*

*Set-Location "$($Path.InstallationPath)"*

*Start-Process "ZWbPe.exe"*

*Start-Process "DMPHelpDesk.exe" -ArgumentList "/iamlive"*

*Start-Process "DMPHelpDesk.exe" -ArgumentList "/uploadstatus"*

*Start-Process "ZWbPe.exe"*

*Write-Output @"*

*`nWebposting binaries has been successfully executed . Binaries details:*

*DMPHelpDesk.exe (Version : $Version)*

*ZWbPe.exe (Version : $Version1)*

*"@*

*}*

*}*

*catch {*

*Write-Output "`n[Exception Error]: $($\_.Exception.message)"*

*}*

*}*

**Task 8**

* **Read Agent communication (DMPHelpDesk) logs.**
* **Path “***"$($Path.InstallationPath)\Applicationlog\DMPHelpDesk.log***”**
* **Logs will have error in below format.**
* **“Date/time 0 0 0 12169 The supplied certificate is invalid." (Regx query used :** *"\d{2,}\/\d{2,}\/\d{2,}\s{1}\d{2}\:\d{2}\:\d{2}\s{1}\w{2}.\*0\s0\s0\s.\*([\d]{5,}.\*)"***)**
* *“***Date/time Err-No : -2147012739***”*

*function Get-logs {*

*try {*

*$ErrorActionPreference = "SilentlyContinue"*

*if ($env:PROCESSOR\_ARCHITECTURE -eq "AMD64") {*

*$Path = (Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\SAAZOD")*

*}*

*elseif ($env:PROCESSOR\_ARCHITECTURE -eq "x86") {*

*$Path = (Get-ItemProperty "HKLM:\SOFTWARE\SAAZOD")*

*}*

*$logs = (Get-Item "$($Path.InstallationPath)\Applicationlog\DMPHelpDesk.log").LastWriteTime*

*if ($logs -ge (Get-Date).AddMinutes(-2)) {*

*$x = $(Get-Content "$($Path.InstallationPath)\Applicationlog\DMPHelpDesk.log" | Select-Object -Last 20)*

*Switch -Regex($x) {*

*"\d{2,}\/\d{2,}\/\d{2,}\s{1}\d{2}\:\d{2}\:\d{2}\s{1}\w{2}.\*0\s0\s0\s.\*([\d]{5,}.\*)" { [array]$Err += $Matches[1] }*

*".\*Err-No : -2147012739" { Write-Output "[Err-No] : 2147012739 `n[Err-Desc] : Unable to Send HTTP Request - An error occurred in the secure channel support, Source : FuncWPWithoutProxy()" ; [array]$Err += $Matches[1] }*

*}*

*if (!$Err)*

*{ Write-Output "[MSG] : No specific Error found under logs. Please login and check the logs, if still issue persists" }*

*else {*

*$Err = $Err | sort-object -Unique*

*write-output $Err*

*}*

*}*

*else {*

*Write-Output @"*

*`nThe logs last write time is older than 2 minutes.*

*Last Write Time : $($logs)*

*"@*

*}*

*}*

*catch {*

*Write-Output "$($\_.Exception.message)"*

*Exit*

*}*

*}*