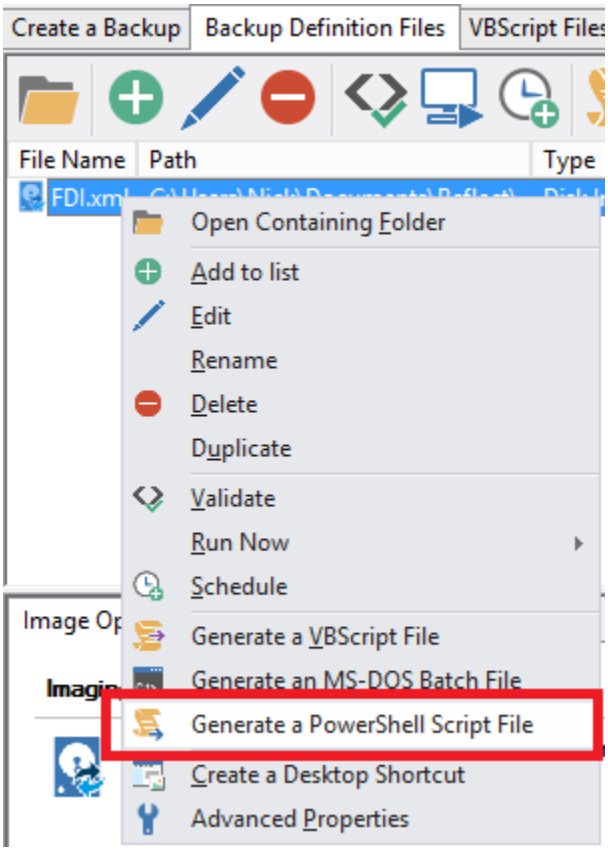
Generating a PowerShell source file

For an example of a generated PowerShell source file please see: Example PowerShell Source File

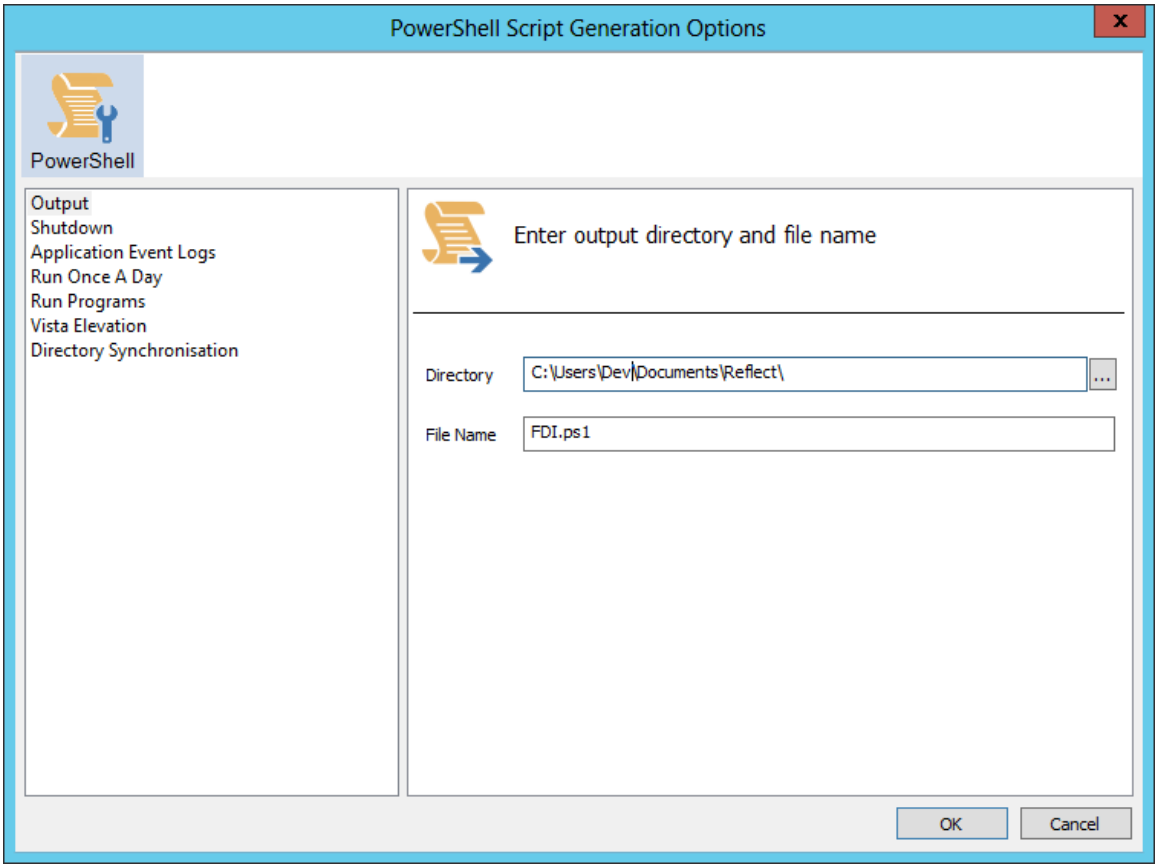
---

To generate a PowerShell file select the 'XML Definition Files' tab on the main window then right click on a file and select **'Generate a PowerShell Script File'**.



This opens the **PowerShell Script Generation Options** dialog:

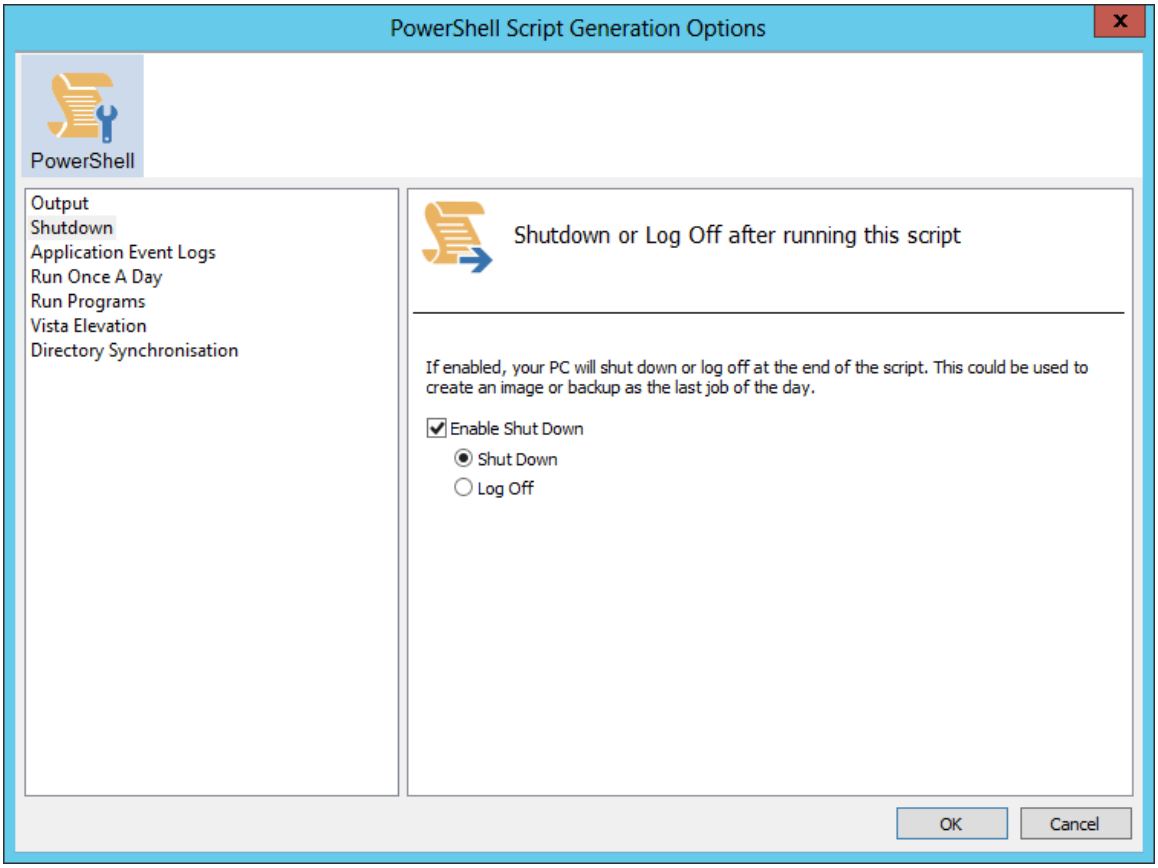
Output



Property	Value
Directory	The folder where the source file is saved
File name	The name of the PowerShell source file. This defaults to the XML file name with a ..ps1 extension

---

Shutdown

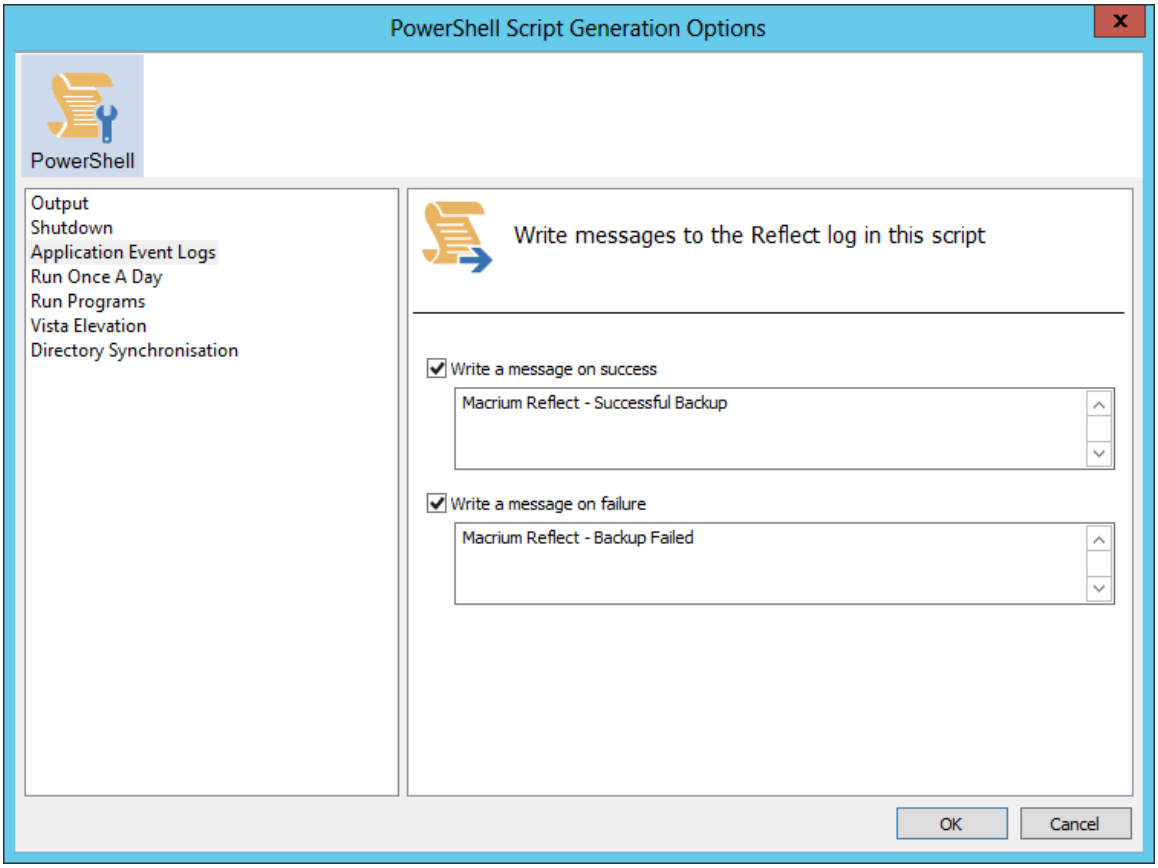



```
Write-Host ' * Initiating shutdown... ' -NoNewLine;
(Get-WMIObject Win32_OperatingSystem -ComputerName '.' -EnableAllPrivileges).Win32Shutdown(8);
```

Property	Value
Enable Shut Down	Select to enable this property
Shut down	The PC will shut own at the end of the backup
Log Off	The user will be logged out at the end of the backup

Application Event logs

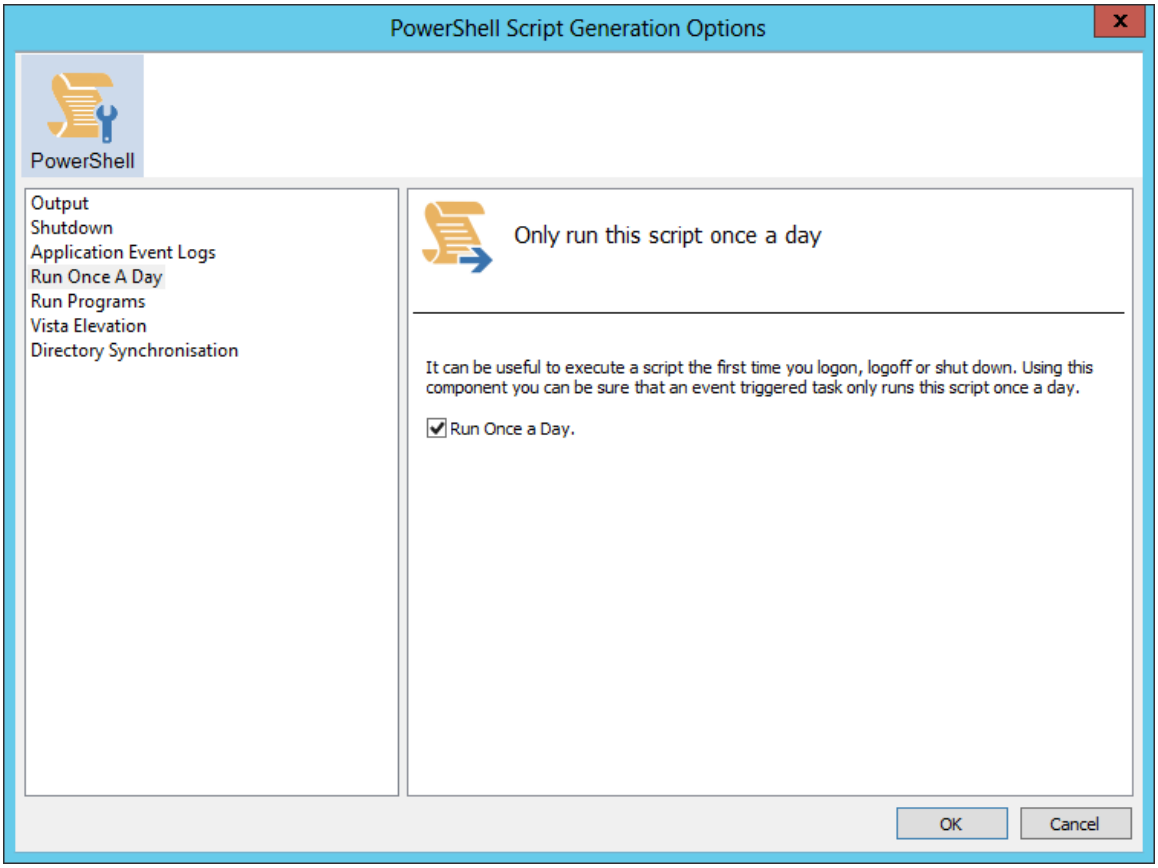
Please note that Macrium Reflect Server and Server Plus Edition includes Windows Event logging built-in.




```
  if ($iExitCode -eq 0) # if backup completed successfully...
  {
    Write-EventLog -EntryType Information -EventId 0 -LogName Application -Message "Macrium Reflect -
Successful Backup" -Source "Macrium Reflect"
  }
  if ($iExitCode -ne 0) # if backup failed...
  {
    Write-EventLog -EntryType Error -EventId 1 -LogName Application -Message "Macrium Reflect - Backup
Failed" -Source "Macrium Reflect"
  }
```

Property	Value
Write a message on success	Enable a Windows Event on successful backup
	Enter the message text in the edit box below
Write a message on failure	Enable a Windows Event for failed backups
	Enter the message text in the edit box below

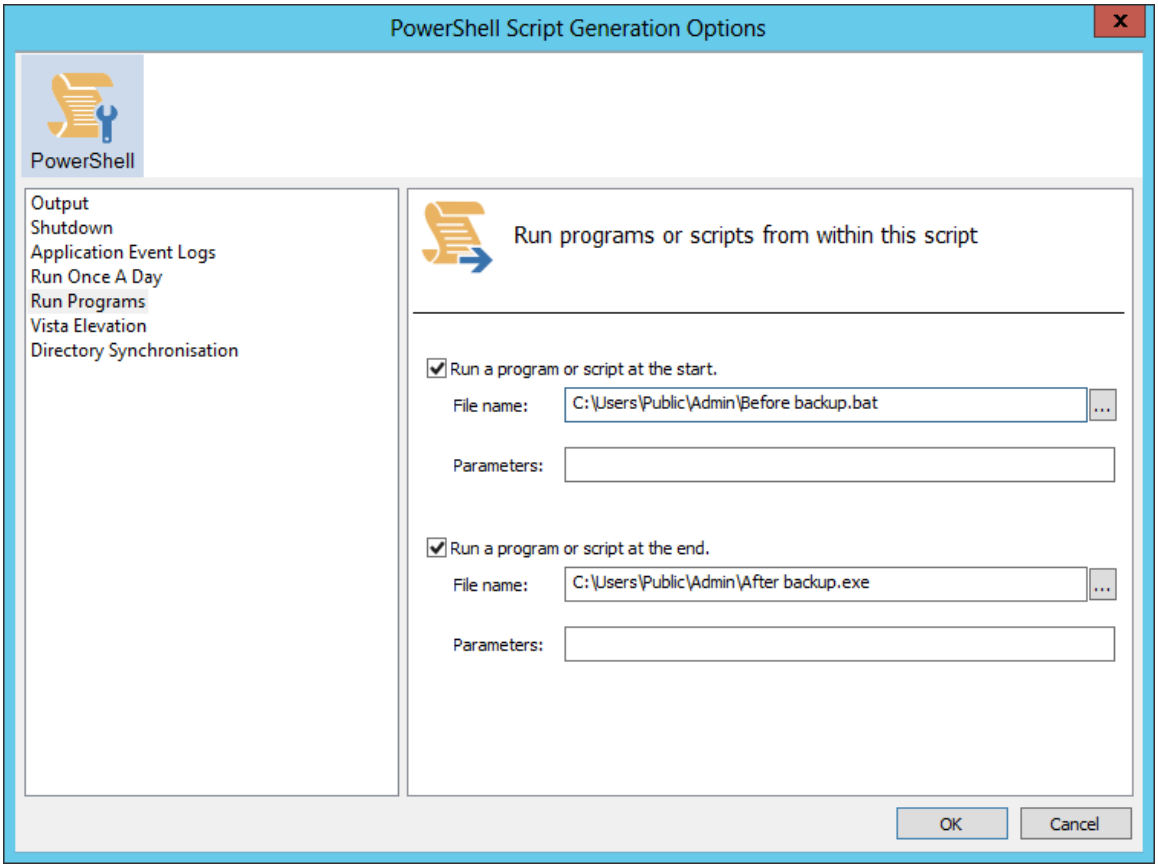
Run Once a Day


PowerShell Script Generation Options
PowerShell
Output
Shutdown
Application Event Logs
Run Once A Day
Run Programs
Vista Elevation
Directory Synchronisation
Only run this script once a day
It can be useful to execute a script the first time you logon, logoff or shut down. Using this component you can be sure that an event triggered task only runs this script once a day.
Run Once a Day.
OK
Cancel

```
  if (HasRunToday)
  {
    Write-Host ' * Script already executed today. Exiting...';
    Write-Host 'Script finished with exit code 0.';
    Exit 0;
  }

..................

#*****************************************************************************
#* Func: HasRunToday
#*
#* Desc: determines if this script has run today
#*
#*****************************************************************************
function HasRunToday()
{
    Write-Host ' * Checking last run time... ' -NoNewLine;
    $strRegPath    = 'HKCU:\Software\Macrium\Reflect\Scripts';
    $boolRanToday  = $false;
    $strDateToday  = Get-Date -UFormat %Y%m%d;
    if (Test-Path $strRegPath)
    {
        try
        {
            $strLastRunDate = (Get-ItemProperty -Path $strRegPath -Name $strScriptPath -ErrorAction
Stop).$strScriptPath
            if ($strLastRunDate -eq $strDateToday)
            {
                $boolRanToday = $true;
            }
        } catch { };
    }
    if (!$boolRanToday)
    {
        Set-ItemProperty -Path $strRegPath -Name $strScriptPath -Value $strDateToday;
    }
    Write-Host 'Done.';
    return $boolRanToday;
}
```

Property	Value
Run Once a Day	If selected, will only enable this script to run once per day. This is useful for example if you want a backup to happen at first login or shutdown.

Run Programs

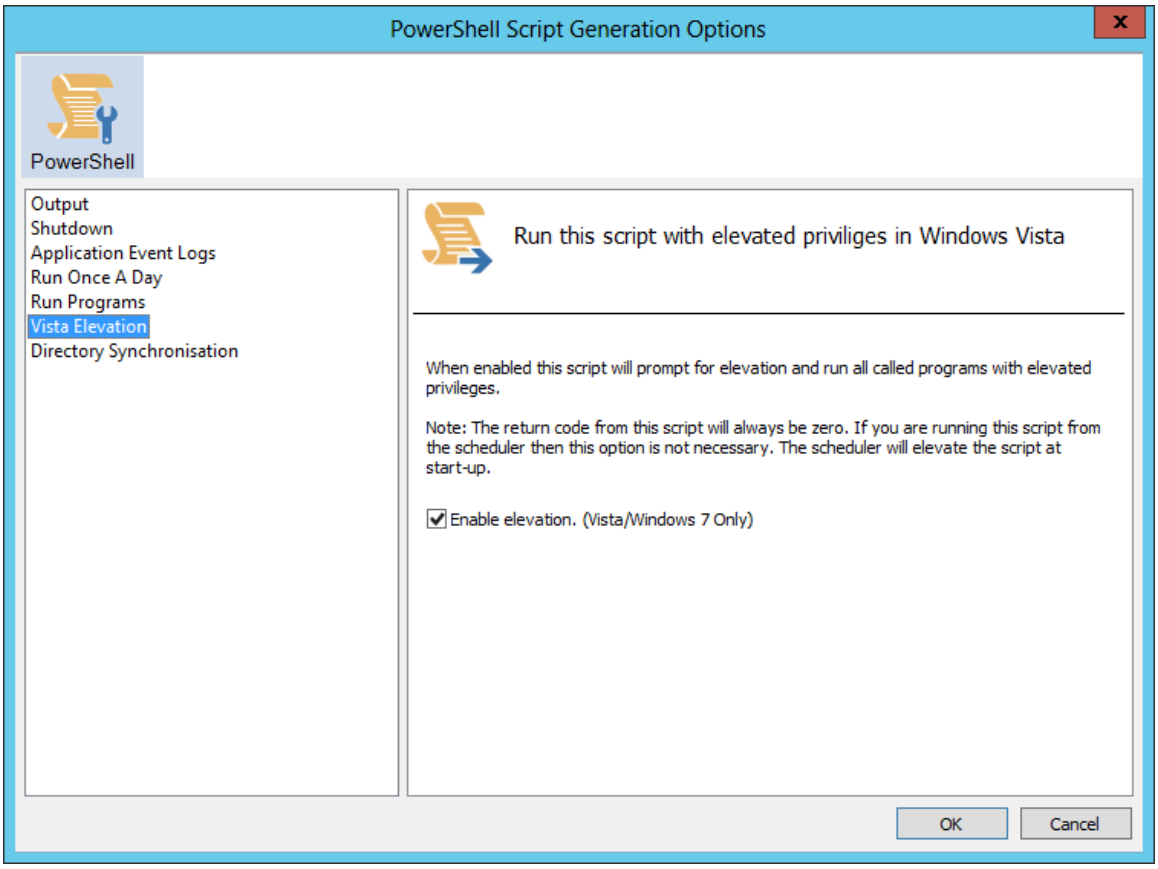



```
  Write-Host ' * Executing "at start" program... ' -NoNewLine;
  $strRunAtStartApp  = 'C:\Users\Public\Admin\Before backup.bat';
  $strRunAtStartArgs = '';
  if ([string]::IsNullOrEmpty($strRunAtStartArgs))
  {
    Start-Process -FilePath $strRunAtStartApp;
  }
  else
  {
    Start-Process -FilePath $strRunAtStartApp -ArgumentList $strRunAtStartArgs;
  }

......

  Write-Host ' * Executing "at end" program... ' -NoNewLine;
  $strRunAtEndApp  = 'C:\Users\Public\Admin\After backup.exe';
  $strRunAtEndArgs = '';
  if ([string]::IsNullOrEmpty($strRunAtEndArgs))
  {
    Start-Process -FilePath $strRunAtEndApp;
  }
  else
  {
    Start-Process -FilePath $strRunAtEndApp -ArgumentList $strRunAtEndArgs;
  }
```

Property	Value
Run a program or script at the start	Select to enable a program or script to run before the backup starts
File name	The path and executable file name

Parameters	Optional command line parameters for the program
Run a program or script at the start	Select to enable a program or script to run when the backup ends
File name	The path and executable file name
Parameters	Optional command line parameters for the program

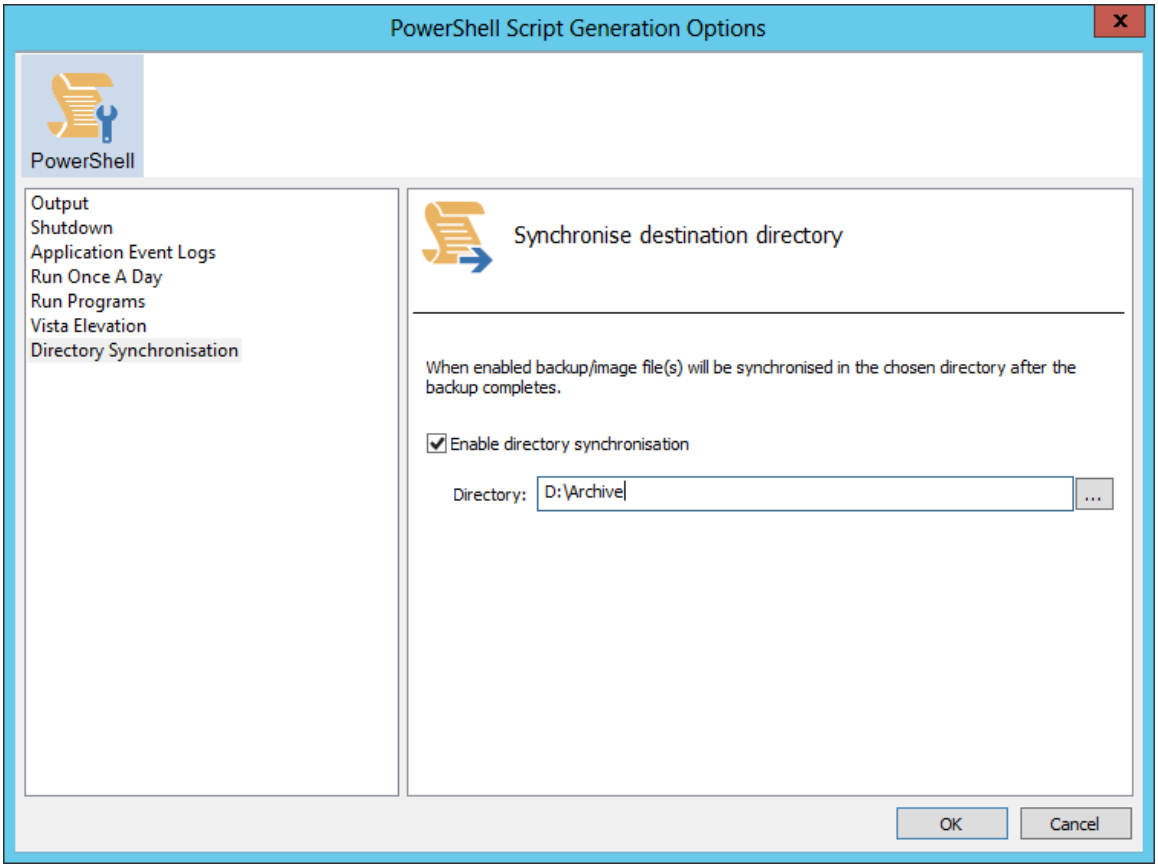
Vista Elevation



```
function Main()
{
  Write-Host 'PowerShell script for Macrium Reflect Backup Definition File';
  Write-Host "BDF: $strXmlFilePath";
  Elevate;

.................


#*******************************************************************************
#* Func: Elevate
#*
#* Desc: Elevates this script for UAC.
#*       This means that only one UAC Elevation prompt is displayed and
#*       functions/programs will not fail if they require admin privileges.
#*
#*******************************************************************************
function Elevate()
{
  # Only elevate if not ran from the task scheduler.
  Write-Host ' * Checking elevated access rights... ' -NoNewLine;
  if (-Not $s)
  {
    # Check to see if we are currently running "as Administrator"
    if (!([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole
([Security.Principal.WindowsBuiltInRole]"Administrator"))
    {
      $ElevatedProcess = new-object System.Diagnostics.ProcessStartInfo "PowerShell";
      # Specify the current script path and name as a parameter
      $strType = GetBackupTypeParameter;
      $ElevatedProcess.Arguments = "-ExecutionPolicy Bypass & '" + $script:MyInvocation.MyCommand.Path + "'
$strType";
      # Indicate that the process should be elevated
      $ElevatedProcess.Verb = "runas";
      # Start the new process
      [System.Diagnostics.Process]::Start($ElevatedProcess);
      # Exit this unelevated script with exit code for "Error: Not elevated"
      Exit 3;
    }
  }
  Write-Host 'Done.';
}
```

Property	Value
Enable Elevation	If selected will enable UAC elevation for the entire script. This enables functions and programs to run outside the context of Macrium Reflect without requesting further elevation.

Directory Synchronization


PowerShell Script Generation Options
PowerShell
Output
Shutdown
Application Event Logs
Run Once A Day
Run Programs
Vista Elevation
Directory Synchronisation
Synchronise destination directory
When enabled backup/image file(s) will be synchronised in the chosen directory after the backup completes.
Enable directory synchronisation
Directory:
D:\Archive
...
OK
Cancel

```
  if ($iExitCode -eq 0) # if backup completed successfully...
  {
    $strBackupDir = GetLastBackupPath;
    if (![string]::IsNullOrEmpty($strBackupDir))
    {
      SynchroniseDirectories $strBackupDir 'D:\Archive';
    }
  }

.................

#*******************************************************************************
#* Func: SynchroniseDirectories
#*
#* Desc: Copies all Macrium Reflect files to a supplied directory.
#*
#*******************************************************************************
function SynchroniseDirectories($strSrcDir, $strDstDir)
{
  Write-Host ' * Synchronising directories... ' -NoNewLine;
  if (Get-Command robocopy -ErrorAction SilentlyContinue)
  {
    # robocopy is available...
    # /copy:DAT - D:Data
    #             A:Attributes
    #             T:Time stamps
    # /purge    - deletes destination files and directories that no longer
    #               exist in the source
    # /lev:0    - Does not copy subdirectories
    &robocopy $strSrcDir $strDstDir *.mr* /copy:DAT /lev:0 /purge /r:0 | Out-Null
  }
  else
  {
    # Fall back to xcopy...

    # Delete files from the target directory not present in the source directory
    $strDstDirChildren = $strDstDir+"\*";
    Get-ChildItem $strDstDirChildren -include "*.mr*" | Foreach-Object {
      $strMaybeDeletedSrcFile = $strSrcDir + '\' + $_.Name;
      if (-not (Test-Path $strMaybeDeletedSrcFile))
      {
        Remove-Item $_
      }
      else
      {
        $SrcFileTime = [datetime](Get-ItemProperty -Path $strMaybeDeletedSrcFile -Name LastWriteTime).
lastwritetime;
        $DstFileTime = [datetime](Get-ItemProperty -Path $_.FullName             -Name LastWriteTime).
lastwritetime;
        $SrcFileTimeString = $SrcFileTime.ToString("yyyMMddHHmmss")
        $DstFileTimeString = $DstFileTime.ToString("yyyMMddHHmmss")
        if ($SrcFileTimeString -ne $DstFileTimeString)
        {
          Remove-Item $_
        }
      }
    }
    # /c - Continues copying even if errors occur.
    # /d - Date check; only copies if file does not exist or is older.
    # /h - Copies hidden and system files.
    # /i - If the destination does not exist, and you are copying more than one
    #        file, this switch assumes that the destination is a folder.
    # /v - Verifies each new file.
    # /y - Overwrites existing files without prompting.
    &xcopy $strSrcDir\*.mr* $strDstDir /c /d /h /i /v /y | Out-Null
  }
  Write-Host 'Done.';
}
```

Property	Value
Enable directory synchronization	If enabled will use the MS utility RoboCopy to synchronize the backup target directory with the supplied directory.
Directory	Enter the folder that you want to synchronize your backup files to

Click **'OK'** to generate the PowerShell source file