_Example_PowerShell_Source_File

Example PowerShell source file generated by Macrium Reflect

See also: Generating a PowerShell source file

```
#*********************************************************************
*****
#*
#* Module Name: Example.ps1
#*
#* Abstract:    This is a template PowerShell Script file generated by
Reflect.
#*              Modify to add your own functionality if required.
#*
#* Returns:     0 - Success.
#*              1 - Error: Invalid XML.
#*              2 - Error: Backup failed.
#*              3 - Error: Not elevated (script will try to execute
elevated).
#*
#*********************************************************************
*****
Param([switch]$s, [switch]$full, [switch]$inc, [switch]$diff)
$strReflectPath = "C:\C++\v6\Image\reflect\x64\Debug\reflect.exe";
$strXmlFilePath = "C:\Example.xml";
$strScriptPath = $MyInvocation.MyCommand.Definition;
#*********************************************************************
*****
#* Func: OnXmlValidationError
#*
#* Desc: Called when a backup fails due to an XML validation error.
#*       This is here to be modified for your own uses.
#*
#*********************************************************************
*****
function OnXmlValidationError()
{
  Write-Warning " ! XML invalid ($strXmlFilePath).";
  # Handle invalid XML error...

}
#*********************************************************************
*****
#* Func: OnBackupError
#*
#* Desc: Called when a backup fails.
#*       This is here to be modified for your own uses.
#*
#*********************************************************************
*****
```

```
function OnBackupError()
{
  Write-Warning " ! Backup failed ($strXmlFilePath).";
  # Handle backup error...

}
#*********************************************************************
*****
#* Func: OnBackupSuccess
#*
#* Desc: Called when a backup succeeds.
#*       This is here to be modified for your own uses.
#*
#*********************************************************************
*****
function OnBackupSuccess()
{
  Write-Host " * Backup succeeded ($strXmlFilePath).";
  # Handle backup success...

}
#*********************************************************************
*****
#* Func: Main
#*
#* Desc: This is main function to start execution.
#*
#*********************************************************************
*****
function Main()
{
  Write-Host 'PowerShell script for Macrium Reflect Backup Definition
File';
  Write-Host "BDF: $strXmlFilePath";
  Elevate;
  $iExitCode = Backup;

  Write-Host "Script finished with exit code $iExitCode.";
  Exit $iExitCode;
}
#*********************************************************************
*****
#* 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.WindowsIdentit
y]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole]"Adminis
trator"))
    {
      $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.';
}
#*******************************************************************
*****
#* Func: Backup
#*
#* Desc: Calls Reflect.exe passing an XML BDF as a parameter.
#*
#*******************************************************************
*****
function Backup()
{
  Write-Host ' * Running the backup... ' -NoNewLine;
  $strType = GetBackupTypeParameter;
  $strArgs = "-e -w $strType `"$strXmlFilePath`"";
  $iResult = (Start-Process -FilePath $strReflectPath -ArgumentList
$strArgs -PassThru -Wait).ExitCode;
  Write-Host 'Done.';
  switch ($iResult)
  {
    2 { OnXmlValidationError; break; }
    1 { OnBackupError;        break; }
    0 { OnBackupSuccess;      break; }
  }
  return $iResult;
}
#*******************************************************************
*****
#* Func: GetBackupTypeParameter
```

```
#*
#* Desc: Determines the backup type from command line parameter...
#*        "-full": Full backup
#*        "-inc" : Incremental backup
#*        "-diff": Differential backup
#*
#*************************************************************************
*****
function GetBackupTypeParameter()
{
  if ($full -eq $true) { return '-full'; };
  if ($inc  -eq $true) { return '-inc';  };
  if ($diff -eq $true) { return '-diff'; };
  return ''; # Clone
```

```
}
# Execute the Main function
Main;
```