Technical white paper

HP Write Manager WMI Scripts

Class Definitions and PowerShell Scripts

Table of contents

Introduction

After installation of HP Write Manager the .mof files containing the WMI Catalog can be found at c:\windows\system32\hpwritefilter.mof and c:\windows\system32\hpdiskcache.mof

Important
To enable HP Write Manager with command-line or WMI commands, the command for Protection must be enabled first. Once the drive is protected, HP Write Manager can be enabled.

Using this guide

The text in `Courier New (font)` under the headings labeled PowerShell Script can be copied and pasted and then saved as PowerShell script (.ps1).

For example:

To add a file exclusion to HP Write Manager

1. Refer to the **Adding a file exclusion to HP Write Manager** section on page 3.
2. Copy and paste the PowerShell Script code save it as .ps1 file. In this example, the file is saved as HPWMAddExclusion.ps1
3. Run `PowerShell.exe.\HPWMAddExclusion.ps1 c:\folder\file_to_exclude.ext`
4. Add the command above as a task to either HP Device Manager, SCCM, or other management/deployment solution.

Commonly used command lines

The following are frequently used features of HP Write Manager.

How to enable or disable HP Write Manager
If it is necessary to make permanent system configuration changes, an administrator can disable HP Write Manager, and then reenable it as necessary. Enabling or disabling requires a system restart.

Click here to be taken to the section on enabling HP Write Manager

Click here to be taken to the section on disabling HP Write Manager

Note
Before enabling HP Write Manager for the first time, you may need to set drive protection. See section **Add a volume to the protected list on page 8** for more information.

How to add or remove a file exclusion
Files listed in exclusion list will be written directly to the flash drive. I

Click here to be taken to the section on adding a file exclusion

Click here to be taken to the section on removing a file exclusion

How to add or remove a registry exclusion
Registries listed in the exclusion list will be written to the flash drive.

Click here to be taken to the section on adding a registry to the exclusion list

Click here to be taken to the section on removing a registry from the exclusion list

How to modfy max size of overlay

Important
By default, the maximum overlay is based on how much memory is available on the system that HP Write Manager is installed onto. Changing the maximum limit for an overlay should only be performed by administrators with experience and knowledge of the specifications and configuration of the system being used. Before making any changes, it is recommended to test image performance to determine the appropriate memory required. Do not increase the maximum overlay limit to a number higher than available system memory.

Click here to be taken to the section on changing the size of the overlay.

Displaying cache usage for a specific file

The class HpWriteFFileCacheUsageInfo displays cache usage for a specific file.

Syntax

```
Class HpWriteFFileCacheUsageInfo {
  [Read] sint64 FileSize;
  [Read] sint64 CacheSize;
  [Read] uint32 OpenHandleCount;
  [Key,Read] string ProcessName;
  [Key] string UserName;
  [Key,Read] stringFileName;
```

Table 1. Property descriptions for HpWriteFFileCacheUsageInfo

Property	Data type	Qualifiers	Description
sint64 FileSize	sint64	[Read]	The size of the file in the file system (in bytes)
sint64 CacheSize	sint64	[Read]	The size of the file in the cache (in bytes).
uint32 OpenHandleCount	uint32	[Read]	Number of open handles to the file.
string ProcessName	string	[Key, Read]	Process that created the file.
string UserName	string	[Key]	User that created the file.
string FileName	string	[Key, Read]	The full name of the file.

Configuring HP Write Manager

HpWriteFConfig configures HP Write Manager. This includes adding exclusions by path, by volume, and by key.

Syntax and property descriptions

```
class HpWriteFConfig {

PropertyContext("Enabled"): ToInstance,
     [Dynamic, Provider ("HpCacheProvider"]
      boolean Enabled;

[PropertyContext("Active"): ToInstance,]
     [Dynamic, Provider("HpCacheProvider"]
      boolean NextEnabled;

[Static] uint32 EnableFilter();
[Static] uint32 DisableFilter();

[Static] uint32 ProtectVolume ([IN] string Path);
[Static] uint32 UnprotectVolume ([IN] string Path);
[Static] uint32 ForceNextSession ([IN] string Path);

[Static] uint32 SetCacheSize
([IN] string Path),
([IN] sint64 CacheSize);

[Static] uint32 SetAlertMarkers
([IN] string Path),
([IN] sint64 RedMarker),
([IN] sint64 YellowMarker),
([IN] sint64 GreenMarker);

[Static] uint32 AddExclusion ([IN] string Path);
[Static] uint32 RemoveExclusion([IN] string Path);

[Expensive, Static] uint32 QueryCacheUsage
([IN] string Path),
([IN] Boolean RecurseSubdirectories),
([OUT] HpWriteFFileCacheUsageInfo FileInformation);}
```

Table 2. Property descriptions for HpWriteFConfig

Property	Data type	Qualifiers	Description
Enabled	boolean	[Read, Dynamic, Provider ("HpCacheProvider")]	This property is TRUE if

			the HP Write Filter is enabled for the current session, or FALSE if disabled.
NextEnabled	boolean	[Read, Dynamic, Provider ("HpCacheProvider")]	Gets the state of the HP Write Filter for the next session. TRUE means it will be enabled in the next session.
RecurseSubdirectories	boolean	[IN]	Sets whether directory paths should also enumerate sub-directories.

Methods for HpWriteFConfig

The following table outlines the methods for the class context HpWriteFConfig. For more information, including PowerShell Scripts, click on the method name within the table.

Table 3. Method descriptions for HpWriteFConfig

Methods	Description
EnableFilter();	Enables HP Write Manager on the next reboot.
DisableFilter();	Disables HP Write Manager on the next reboot.
ProtectVolume(string Path);	Adds a volume to the protected list for the next session.
UnprotectVolume(string Path);	Removes a volume to the protected list for the next session.
ForceNextsesson(string Path);	Forces the next session to immediately take effect without a reboot.
SetCacheSize(string Path, sint64 CacheSize);	Sets the cache size for the protected volume on reboot.
SetAlertMarkers(string Path, sint64 RedMarker, sint64 YellowMarker, sint64 GreenMarker);	Sets the red, yellow, and return to green marker levels for the protected volume.
AddExclusion (string Path);	Adds a file or directory to the file exclusion list for the next session.
RemoveExclusion (string Path);	Removes a file or directory from the file exclusion list for the next session.
QueryCacheUsage([IN] string Path, [IN] boolean RecurseSubdirectories, [OUT] HpWriteFFileCacheUsageInfo FileInformation);	Returns the cache usage for a file or directory of files. The length of this operation may vary depending on the number of files queried

Enable HP Write Manager on next reboot
Use the method HpWriteFConfig.EnableFilter to enable HP Write Manager on the next reboot.

Note
Before enabling HP Write Manager for the first time, you may need to set drive protection. See section **Add a volume to the protected list on page 8** for more information.

Syntax

```
uint32 EnableFilter();
```

Return Value

Returns 0 when successful. Otherwise, it returns error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
$Ret=$HpWF.EnableFilter()

if ($Ret.ReturnValue -gt 0)
{
write-Host "Enabling filter failed with error" $Ret.ReturnValue
}
else
{
write-Host "Enabling filter succeeded!"
}
```

Disable Hp Write Manager on next reboot
Use Hp.WriteFConfig.DisableFilter to disable HP Write Manager on the next reboot.

Syntax

```
uint32 DisableFilter();
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
$Ret=$HpWF.DisableFilter()
```

```
if ($Ret.ReturnValue -gt 0)
{
write-Host "Disabling filter failed with error" $Ret.ReturnValue
}
else
{
write-Host "Disabling filter succeeded!"
}
```

Add a volume to the protected list

The class HpWriteFConfig.ProtectVolume adds a volume to the protected list for the next session.

Syntax

```
uint32 ProtectVolume ([IN] string Path);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$DriveLetter=$args[0]
if ($DriveLetter)
{
   $HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
   $Ret=$HpWF.ProtectVolume($DriveLetter+":")

   if ($Ret.ReturnValue -gt 0)
   {
      write-Host "Enabling protection on volume "$DriveLetter": failed with error"
$Ret.ReturnValue
   }
   else
   {
      write-Host "protection on volume "$DriveLetter": succeeded!"
   }
}
```

Remove a volume from the protected list

The class HpWriteFConfig.UnprotectVolume removes a volume to the protected list for the next session.

Syntax

```
uint32 UnprotectVolume ([IN] string Path);
```

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$DriveLetter=$args[0]
if ($DriveLetter)
{
$HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
$Ret=$HpWF.UnprotectVolume($DriveLetter+":")

if ($Ret.ReturnValue -gt 0)
{
write-Host "Unprotecting volume "$DriveLetter": failed with error"
$Ret.ReturnValue
}
else
{
write-Host "Unprotecting volume" $DriveLetter": succeeded!"
}
}
```

Force next session without rebooting

The class HpWriteFConfig.ForceNextSession forces the next session to immediately take effect without a reboot.

Syntax

```
uint32 ForceNextSession ([IN] string Path);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$DriveLetter=$args[0]
if ($DriveLetter)
{
$HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
$Ret=$HpWF.ForceNextSession($DriveLetter+":")

if ($Ret.ReturnValue -gt 0)
{
write-Host "Force next for volume "$DriveLetter": failed with error"
$Ret.ReturnValue
}
else
```

```
{
write-Host "Force next for volume" $DriveLetter": succeeded!"
}
}
else
{
```

Set the protected volume's cache size (max overlay size)
The class HpWriteFConfig.SetCacheSize sets the cache size for the protected volume on reboot.

Syntax

```
uint32 SetCacheSize
([IN] string Path),
([IN] sint64 CacheSize);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$DriveLetter=$args[0]
$CacheSize=[Convert]::ToInt64($args[1])

if ($CacheSize)
{
$HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
$Ret=$HpWF.SetCacheSize($DriveLetter+":",$CacheSize)

if ($Ret.ReturnValue -gt 0)
{
write-Host "Setting max cache size on volume "$DriveLetter": failed with error"
$Ret.ReturnValue
}
else
{
write-Host "Setting max cache size on volume "$DriveLetter": succeeded!"
}
}
```

Marker levels for the protected volume
The class HpWriteFConfig.SetAlertMarkers sets the red, yellow, and return to green marker levels for the protected volume.

Syntax

```
uint32 SetAlertMarkers
([IN] string Path),
([IN] sint64 RedMarker),
```

```
([IN] sint64 YellowMarker),
([IN] sint64 GreenMarker);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$DriveLetter=$args[0]
$RedMarker=[Convert]::ToInt64($args[1])
$YellowMarker=[Convert]::ToInt64($args[2])
$GreenMarker=[Convert]::ToInt64($args[3])

if ($GreenMarker)
{
   $HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"

$Ret=$HpWF.SetAlertMarkers($DriveLetter+":",$RedMarker,$YellowMarker,$GreenMarker)

   if ($Ret.ReturnValue -gt 0)
   {
      write-Host "Setting alert markers on volume "$DriveLetter": failed with
error" $Ret.ReturnValue
   }
   else
   {
      write-Host "Setting alert markers on volume "$DriveLetter": succeeded!"

      $Props = Get-WmiObject -namespace root\hp\writefilters
HpWriteFProtectedVolumeInfo

      write-Host "Yellow Marker Cache Size - once hit purging begins: "
$Props.YellowMarker
      write-Host "Red Marker Cache Size - once hit aggressive purging begins: "
$Props.RedMarker
      write-Host "Green Marker Cache Size - once hit purging ceases: "
$Props.GreenMarker
   }
}
```

Adding a file exclusion to HP Write Manager

The class HpWriteFConfig.AddExclusion adds a file or directory to the file exclusion list for the next session.

Syntax

```
uint32 AddExclusion ([IN] string Path);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$ExclusionPath=$args[0]

if ($ExclusionPath)
{

   $HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
   $Ret=$HpWF.AddExclusion($ExclusionPath)

   if ($Ret.ReturnValue -gt 0)
   {
      write-Host "Exclusion path "$ExclusionPath "was NOT added.  Error: "
$Ret.ReturnValue
   }
   else
   {
      write-Host "Exclusion path "$ExclusionPath "added successfully."
   }
}
```

Remove a file or directory from the file exclusion list

The class HpWriteFConfig.RemoveExculsion removes a file or directory from the file exclusion list on the next session

Syntax

```
uint32 RemoveExclusion;
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$ExclusionPath=$args[0]

if ($ExclusionPath)
{
   $HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
   $Ret=$HpWF.RemoveExclusion($ExclusionPath)

   if ($Ret.ReturnValue -eq 0)
   {
```

```
      write-Host "Exclusion path "$ExclusionPath "removed successfully."
$Ret.ReturnValue
   }
   else
   {
   if ($Ret.ReturnValue -eq 2147749890)
      {
         write-Host "Exclusion path "$ExclusionPath "not found.  Error: "
$Ret.ReturnValue
      }
      else
      {
         write-Host "Exclusion path "$ExclusionPath "NOT removed.  Error: "
$Ret.ReturnValue
      }
   }

}
```

See cache usage of file or file directory

The class HpWriteFConfig.QueryCacheUsage returns the cache usage for a file or directory of files.

Note

The length of this operation may vary depending on the number of files queried.

Syntax

```
uint32 QueryCacheUsage
([IN] string Path),
([IN] Boolean RecurseSubdirectories),
([OUT] HpWriteFFileCacheUsageInfo FileInformation);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
QueryPath=$args[0]
$RecurseArg=$args[1]
if (!$RecurseArg)
{
  $RecurseArg=0
}
```

```
if ($QueryPath)
{

#  Check to be sure there are files in the path

   $PathExists = test-path -path $QueryPath

   if ($PathExists)
   {
   $HpWF = [WmiClass]"root\hp\writefilters:HpWriteFConfig"
   $Ret=$HpWF.QueryCacheUsage($QueryPath,$RecurseArg)

   if ($Ret.ReturnValue -gt 0)
   {
      write-Host "Failed to query path "$QueryPath "for cache usage.  Error: "
$Ret.ReturnValue
   }
   else
   {
      write-Host "Query path "$QueryPath "for cache usage succeeded."

      foreach ($x in $Ret.FileInformation)
      {
  write-Host " "
  write-Host "File name: " $x.FileName
  write-Host "File size: " $x.FileSize
  write-Host "Cache size: " $x.CacheSize
  write-Host "Number of handles open: " $x.OpenHandleCount
  write-Host "Process that created file: " $x.ProcessName
  write-Host "User that created file: " $x.UserName
      }

   }
   }
   else
   {
       write-Host "Path does not exist"
   }
}
```

Volume status

HpWriteFVolumeInfo is used to get information about the status of a volume.

Syntax and property descriptions

```
class HpWriteFVolumeInfo {
  [Key, Read] string VolumeName;
  [Key, Read] string DriveLetter;
  [Read] boolean Protected;
  [Read] Boolean NextProtected;
  [Read] stint64 MaximumCacheSize;

[PropertyContext("NextMaximumCacheSize"): ToInstance,]
     [Dynamic, Provider("HpCacheProvider" ]
      sint64 NextMaximumCacheSize;

[PropertyContext("RedMarker"): ToInstance]
     [Dynamic, Provider("HpCacheProvider"): ToInstance]
      sint64 YellowMarker;

[PropertyContext("RedMarker"): ToInstance]
     [Dynamic, Provider("HpCacheProvider"): ToInstance]
      sint64 RedMarker;

[PropertyContext("RedMarker"): ToInstance]
     [Dynamic, Provider("HpCacheProvider"): ToInstance]
      sint64 GreenMarker;

}
```

Table 4. Property descriptions for HpWriteFVolumeInfo

Property	Data type	Qualifiers	Description
VolumeName	string	[Key, Read]	Gets the name of the volume.
DriveLetter	string	[Key, Read]	Gets the currently mapped drive letter for this volume. Empty if the volume is not mapped to a drive letter
Protected	boolean	[Read]	Gets the current protection state of the volume.

NextProtected	boolean	[Read]	Gets the protection state of the volume for the next session
MaximumCacheSize	sint64	[Read]	Gets the protection state of the volume for the next session
NextMaximumCacheSize	sint64	[Dynamic, Provider("HPCacheProvider")]	Reports the maximum possible cache usage for this session.
YellowMarker	sint64	[Dynamic, Provider("HPCacheProvider")]	Reports the yellow alert marker. Once remaining cache space hits this level, purging of the cache will begin
RedMarker	sint64	[Dynamic, Provider("HPCacheProvider")]	Reports the red alert marker. Once remaining space hits this level, aggressive purging of the cache will begin.
GreenMarker	sint64	[Dynamic, Provider("HPCacheProvider")]	Reports the return to green alert marker. Once remaining space returns to this level, purging of the cache will cease.

Protected volume status information

HpWriteFProtectedVolumeInfo is used to configure, and get information about the status of, a protected volume.

Syntax and property descriptions

```
class HpWriteFVolumeInfo {
  [Key, Read] string VolumeName;
  [Key, Read] string DriveLetter;
  [Read] uint32 SequenceNumber;

PropertyContext("CurrentCacheUsage"): ToInstance,
  [Dynamic, Provider("HpCacheProvider")]
  sint64 CurrentCacheUsage;

PropertyContext("TotalWriteBytes"): ToInstance,
   [Dynamic, Provider("HpCacheProvider")]
   sint64 TotalWriteBytes;

PropertyContext("MetadataWriteBytes"): ToInstance,
   [Dynamic, Provider("HpCacheProvider")]
    sint64 MetadataWriteBytes;
```

```
PropertyContext("CopyOnWriteBytes"): ToInstance,
   [Dynamic, Provider("HpCacheProvider")]
   sint64 CopyOnWriteBytes;
 }
```

Table 5. Property descriptions for HpWriteFProtectedVolumeInfo

Property	Data type	Qualifiers	Description
VolumeName	string	[Key, Read]	Gets the name of the volume.
DriveLetter	string	[Key, Read]	Gets the currently mapped drive letter for this volume. Empty if the volume is not mapped to a drive letter
SequenceNumber	uint32	[Read]	The sequence number of the specified protected volume instance
CurrentCacheUsage	sint64	[PropertyContext]	Reports the current cache usage for the current session.
TotalWriteBytes	sint64	[PropertyContext]	Reports the total number of write bytes seen to the cache (includes overwrites).
MetadataWriteBytes	sint64	[PropertyContext]	Reports the number of bytes used in the cache for NTFS metadata (excludes overwrites).
CopyOnWriteBytes	sint64	[PropertyContext]	Reports the number of bytes used in the cache for writes to the protected volume (excludes overwrites).

Configuring HP Registry Filter

HpRegFConfig is used to configure the HP Registry Filter. This also includes adding exclusions by key.

Syntax and property descriptions

```
class HpRegFConfig {
PropertyContext("GlobalBypassEnabled" : ToInstance,
  [Dynamic, Provider("HpCacheProvider")]
        boolean GlobalBypassEnabled;

PropertyContext("GlobalNextBypassEnabled" : ToInstance,
  [Dynamic, Provider("HpCacheProvider")]
```

```
        boolean GlobalNextBypassEnabled;

[Static] uint32 EnableGlobalBypass();
[Static] uint32 UpdateGlobalBypass();
[Static] uint32 AddExclusion([IN] string Path);
[Static] uint32 RemoveExclusion ([IN] string Path);
[Static] void UpdateExclusions();
```

Table 6. Property descriptions for HpRegFConfig

Property	Data type	Qualifiers	Description
GlobalBypassEnabled	boolean	[PropertyContext]	Reports the global bypass state of the registry filter. TRUE means all registry paths are excluded and any registry modifications persist.
GlobalBypassNextEnabled	boolean	[PropertyContext]	Reports the global bypass state of the registry filter the next time the system is restarted ot when UpdateGlobalBypass is called.

Methods for HpRegFConfig

The following table outlines the methods for the class context HpRegFConfig. For more information, including PowerShell Scripts, click on the method name within the table.

Table 7. Method descriptions for HpRegFConfig

Methods	Description
EnableGlobalBypass()	Enables global registry bypass the next time the system is restarted, or when UpdateGlobalBypass is called.
DisableGlobalBypass();	Disables global registry bypass the next time the system is restarted, or when UpdateGlobalBypass is called.
UpdateGlobalBypass();	Causes Global Bypass to be immediately enabled or disabled, based on GlobalBypassNextEnabled.
AddExclusion(string Path);	Adds a key (and all values/subkeys of that key) to the key exclusion list. The path must be a path beginning with a well-known key (e.g. HKLM\\System\\CurrentControlSet\\Services or HKU\\.DEFAULT).
RemoveExclusion(string Path);	Removes a key from the key exclusion list. The path must be a path beginning with a well known key(e.g. HKLM\\System\\CurrentControlSet\\Services, HKU\\.DEFAULT).
UpdateExlusions();	Makes all NextActive key exclusions become

active.

Enable global registry bypass

The class HpRegFConfig enables global registry bypass the next time the system is restarted, or when UpdateGlobalBypass is called.

Syntax

```
[Static] uint32 EnableGlobalBypass();
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$HpRF = [WmiClass]"root\hp\writefilters:HpRegFConfig"
$Ret=$HpRF.EnableGlobalBypass()

if ($Ret.ReturnValue -gt 0)
{
   write-Host "Enabling global bypass for registry failed with error"
$Ret.ReturnValue
}
else
{
   write-Host "Enabling global bypass for registry succeeded!"

#  Be sure properties are set correctly
   $Props = Get-WmiObject -namespace root\hp\writefilters HpRegFConfig
   write-Host "Enabled flag: " $Props.GlobalBypassEnabled
   write-Host "NextEnabled flag: " $Props.GlobalBypassNextEnabled
   if ($Props.GlobalBypassNextEnabled)
   {
      write-Host "Global bypass will be enabled in next session"
   }

}
```

Disable global registry bypass

HpRegFConfig.DisablesGlobalBypass, disables registry bypass the next time the system is restarted, or when UpdateGlobalBypass is called.

Syntax

```
[Static] uint32 DisableGlobalBypass();
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$HpRF = [WmiClass]"root\hp\writefilters:HpRegFConfig"
$Ret=$HpRF.DisableGlobalBypass()



if ($Ret.ReturnValue -gt 0)
{
   write-Host "Disabling global bypass for registry failed with error"
$Ret.ReturnValue
}
else
{
   write-Host "Disabling global bypass for registry succeeded!"

#  Be sure properties are set correctly
   $Props = Get-WmiObject -namespace root\hp\writefilters HpRegFConfig
   write-Host "Enabled flag: " $Props.GlobalBypassEnabled
   write-Host "NextEnabled flag: " $Props.GlobalBypassNextEnabled
   if (!$Props.GlobalBypassNextEnabled)
   {
      write-Host "Global bypass will be disabled in next session"
   }



}
```

Immediately enable or disable global bypass

HpRegFConfig.UpdateGlobalBypass causes global registry bypass to be immediately enabled or disabled, based on GlobalBypassNextEnabled.

Syntax

```
[Static] uint32 UpdateGlobalBypass();
```

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
HpRF = [WmiClass]"root\hp\writefilters:HpRegFConfig"
$Ret=$HpRF.UpdateGlobalBypass()


if ($Ret.ReturnValue -gt 0)
{
   write-Host "Global bypass state for registry failed to be updated with error"
$Ret.ReturnValue
```

```
}
else
{
   write-Host "Global bypass for registry has been updated!"


#  Be sure properties are set correctly
   $Props = Get-WmiObject -namespace root\hp\writefilters HpRegFConfig
   write-Host "Enabled flag: " $Props.GlobalBypassEnabled
   write-Host "NextEnabled flag: " $Props.GlobalBypassNextEnabled
   if ($Props.GlobalBypassNextEnabled)
   {
      write-Host "Global bypass will be enabled in next session"
   }

}
```

Add a key to exclusion list

HpRegFConfig.AddExclusion adds a key (and all values/subkeys of that key) to the key exclusion list. The path must be a path beginning with a well-known key (e.g. HKLM\\System\\CurrentControlSet\\Services or HKU\\.DEFAULT).

Syntax

```
[Static] uint32 AddExclusion([IN] string Path);
```

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$ExclusionKey=$args[0]

if ($ExclusionKey)
{
   $HpRF = [WmiClass]"root\hp\writefilters:HpRegFConfig"
   $Ret=$HpRF.AddExclusion($ExclusionKey)

if ($Ret.ReturnValue -gt 0)
{
   write-Host "Exclusion key "$ExclusionKey "was NOT added.  Error: "
$Ret.ReturnValue
}
else
{
   write-Host "Exclusion key"$ExclusionKey "added successfully."


#  Be sure properties are set correctly
```

```
   $Props = Get-WmiObject -namespace root\hp\writefilters HpRegFExclusionEntry

   write-Host " "
   write-Host "Active Key Exclusions"
   write-Host " "

   foreach ($x in $Props)
   {
         if ($x.Active)
         {
            write-Host "Excluded Key: " $x.KeyName
         }
   }

   write-Host " "
   write-Host "Key Exclusions in Next Session"
   write-Host " "

   foreach ($x in $Props)
   {
         if ($x.NextActive)
         {
            write-Host "Excluded Key: " $x.KeyName
         }
   }

}
}
else
{
   write-Host "HPAddExclusion.ps1 needs registry key path to be excluded."
   write-Host "For example:  PowerShell .\HPReg_AddExclusion.ps1
\REGISTRY\MACHINE\SOFTWARE"

}
```

Remove a key from exclusion list

Use HpRegFConfig to removes a key from the key exclusion list. The path must be a path beginning with a well-known key (e.g. HKLM\\System\\CurrentControlSet\\Services, HKU\\.DEFAULT).

Syntax

```
[Static] uint32 RemoveExclusion([IN] string Path);
```

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$ExclusionKey=$args[0]

if ($ExclusionKey)
{
   $HpRF = [WmiClass]"root\hp\writefilters:HpRegFConfig"
   $Ret=$HpRF.RemoveExclusion($ExclusionKey)


if ($Ret.ReturnValue -eq 0)
{
   write-Host "Exclusion key"$ExclusionKey "removed successfully."


#  Be sure properties are set correctly
   $Props = Get-WmiObject -namespace root\hp\writefilters HpRegFExclusionEntry

   write-Host " "
   write-Host "Active Key Exclusions"
   write-Host " "

   foreach ($x in $Props)
   {
         if ($x.Active)
         {
            write-Host "Excluded Key: " $x.KeyName
         }
   }

   write-Host " "
   write-Host "Key Exclusions in Next Session"
   write-Host " "

   foreach ($x in $Props)
   {
         if ($x.NextActive)
         {
            write-Host "Excluded Key: " $x.KeyName
         }
   }
```

```
else
{
   if ($Ret.ReturnValue -eq 2147749890)
{
   write-Host "Exclusion key "$ExclusionKey "not found.  Error: " $Ret.ReturnValue
}
else
{
   write-Host "Exclusion key "$ExclusionKey "was NOT removed.  Error: "
$Ret.ReturnValue
}
}

}
else
{
   write-Host "HPREmoveExclusion.ps1 needs registry key path to be removed."
   write-Host "For example:  PowerShell .\HPReg_RemoveExclusion.ps1
\REGISTRY\MACHINE\SOFTWARE"
}
```

Activate NextActive key exclusions

Use HpRegFConfig to makes all NextActive key exclusions become active.

Syntax

```
[Static] uint32 UpdateExclusions()
```

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$HpRF = [WmiClass]"root\hp\writefilters:HpRegFConfig"
$Ret = $HpRF.UpdateExclusions()

if ($Ret.ReturnValue -gt 0)
{
   write-Host "Pending registry exclusions not updated.  Error: " $Ret.ReturnValue
}
else
{
   write-Host "Pending registry exclusions  now active"

#  Be sure properties are set correctly
   $Props = Get-WmiObject -namespace root\hp\writefilters HpRegFExclusionEntry
```

```
    write-Host " "
    write-Host "Active Key Exclusions"
    write-Host " "

    foreach ($x in $Props)
    {
          if ($x.Active)
          {
             write-Host "Excluded Key: " $x.KeyName
          }
    }

    write-Host " "
    write-Host "Key Exclusions in Next Session"
    write-Host " "

    foreach ($x in $Props)
    {
          if ($x.NextActive)
          {
             write-Host "Excluded Key: " $x.KeyName
          }
    }

}
```

Information for files and directories in exclusion list

HpWriteFExclusionEntry contains information about files or directories in the exclusion list of a volume protected by HP Write Filter.

Syntax and property descriptions

```
class HpRegFExclusionEntry {
  [Key, Read] string KeyName;
  [Read] boolean Active;
  [Read] boolean NextActive;
}
```

Table 8. Property descriptions for HpRegFExclusionEntry

Property	Data type	Qualifiers	Description
KeyName	string	[Key, Read]	Gets the name of the volume.
Active	boolean	[Read]	Gets the currently mapped drive letter for

			this volume. Empty if the volume is not mapped to a drive letter
NextActive	boolean	[Read]	The sequence number of the specified protected volume instance

Hp Cache configuration

The class HpCacheDiskConfig is used to configure HP Cache's behavior for a given disk.

Syntax and Property Descriptions

```
class HpCacheDiskConfig {

[Static] uint32 EnableCaching
([IN] uint32 DiskNumber);

[Static] uint32 DisableCaching
([IN] uint32 DiskNumber);

[Static] uint32 SetAllocationPolicy
([IN] uint32 DiskNumber),
([IN] string AllocationPolicy);

[Static] uint32 SetCacheSize
([IN] uint32 DiskNumber),
([IN] sint64 CacheSize);

[Static] uint32 SetFlushTimeInterval
([IN]uint32 DiskNumber)'
([IN]uint32 FlushTimeInterval);

[Static] uint32 SetFlushCount
([IN] uint32 DiskNumber),
([IN] uint32 FlushCount);
}
```

Methods and arguments for HpCacheDiskConfig

The following table outlines the methods for the class context HpCacheDiskConfig. For more information, including PowerShell Scripts, click on the Method name within the table.

Table 9. Metha descriptions for HpCacheDiskConfig

Method (Argument)	Description
EnableCaching(uint32 DiskNumber);	Enables caching on the specified disk.
DisableCaching(uint32 DiskNumber);	Disables caching on the specified disk.
SetAllocationPolicy(uint32 DiskNumber, string AllocationPolicy);	Sets the cache allocation policy for a specified disk. AllocationPolicy can be set to static or dynamic.
SetCacheSize(uint32 DiskNumber, sint64 CacheSize)	Sets the maximum size, in MB, that the cache can grow to. If the AllocationPolicy is 'Static', the cache will be instantiated at this size.
SetFlushTimeInterval(uint32 DiskNumber, uint32 FlushTimeInterval);	Sets the flush interval of the cache. Data will not remain in the cache for longer than the flush time interval specified in seconds.
SetFlushCount(uint32 DisckNumber, uint32 FlushCount)	Sets the number of system flush operations that the flush operation can occur before causing a flush of the HpCache cache.

EnableCaching
Enables caching on the specified disk

Syntax

```
[Static] uint32 EnableCaching
([IN] uint32 DiskNumber);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$Disk=$args[0]
if (!$Disk)
{
  $Disk=0
}
$HpDC = [WmiClass]"root\hp\writefilters:HpCacheDiskConfig"
$Ret=$HpDC.EnableCaching($Disk)

if ($Ret.ReturnValue -gt 0)
{
```

```
   write-Host "Caching NOT enabled on disk"$Disk "- Error:" $Ret.ReturnValue
}
else
{
   write-Host "Caching enabled on disk"$Disk

   $Props = Get-WmiObject -namespace root\hp\writefilters HpCacheDiskInfo
   foreach ($x in $Props) {
      if ($x.DiskNumber -eq $Disk)
      {
         write-Host "Caching set to" $x.CachingEnabled "for disk" $x.DiskNumber
"on next boot"
      }
   }
}
```

DisableCaching
Disable caching on the specified disk

Syntax

```
[Static] uint32 DisableCaching
([IN] uint32 DiskNumber);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$Disk=$args[0]
if (!$Disk)
{
  $Disk=0
}

$HpDC = [WmiClass]"root\hp\writefilters:HpCacheDiskConfig"
$Ret=$HpDC.DisableCaching($Disk)

if ($Ret.ReturnValue -gt 0)
{
   write-Host "Caching NOT disabled on disk"$Disk "- Error:" $Ret.ReturnValue
}
else
```

```
{
   write-Host "Caching disabled on disk"$Disk

   $Props = Get-WmiObject -namespace root\hp\writefilters HpCacheDiskInfo
   foreach ($x in $Props) {
      if ($x.DiskNumber -eq $Disk)
      {
         write-Host "Caching set to" $x.CachingEnabled "for disk" $x.DiskNumber
"on next boot"
      }
   }
}
```

SetAllocationPolicy

Sets the cache allocation policy for a specified disk

Syntax

```
[Static] uint32 SetAllocationPolicy
([IN] uint32 DiskNumber),
([IN] string AllocationPolicy);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$Disk=$args[0]
if (!$Disk)
{
  $Disk=0
}
$APolicy=$args[1]

if ($APolicy)
{

$HpDC = [WmiClass]"root\hp\writefilters:HpCacheDiskConfig"
$Ret=$HpDC.SetAllocationPolicy($Disk,$APolicy)

if ($Ret.ReturnValue -gt 0)
{
   write-Host "Allocation policy NOT set on disk" $Disk "- Error:"
$Ret.ReturnValue
```

```
}
else
{
   write-Host "Allocation policy set on disk" $Disk "to" $APolicy
}


}
else
{
write-Host "HPDC_SetAllocaitonPolicy.ps1 needs disk number and allocation policy
(static or dynamic)"
write-Host "For Example:  PowerShell .\HPDC_SetAllocaitonPolicy.ps1 0 dynamic"
}
```

SetCacheSize

Sets the maximum size, in MB, that the cache can grow to. If the AllocationPolicy is 'Static', the cache will be instantiated at this size.

Syntax

```
[Static] uint32 SetCacheSize
([IN] uint32 DiskNumber),
([IN] sint64 CacheSize);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$Disk=$args[0]
if (!$Disk)
{
  $Disk=0
}
$Size=$args[1]

if ($Size)
{

$CacheSize=[Convert]::ToInt64($Size)

$HpDC = [WmiClass]"root\hp\writefilters:HpCacheDiskConfig"
$Ret=$HpDC.SetCacheSize($Disk,$CacheSize)
```

```
if ($Ret.ReturnValue -gt 0)
{
   write-Host "Cache size NOT set on disk" $Disk "- Error:" $Ret.ReturnValue
}
else
{
   write-Host "Cache size set to" $CacheSize "MB on disk" $Disk "on next boot"
}

}
else
{
write-Host "HPDC_SetCacheSize.ps1 needs disk number and cache size in MB"
write-Host "For Example:  PowerShell .\HPDC_SetCacheSize.ps1 0 10"
}
```

SetFlushTimeInterval

Sets the cache allocation policy for a specified disk

Syntax

```
[Static] uint32 SetFlushTimeInterval
([IN]uint32 DiskNumber)'
([IN]uint32 FlushTimeInterval);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$Disk=$args[0]
if (!$Disk)
{
  $Disk=0
}
$TimeInt=$args[1]

if ($TimeInt)
{
$HpDC = [WmiClass]"root\hp\writefilters:HpCacheDiskConfig"
$Ret=$HpDC.SetFlushTimeInterval($Disk,$TimeInt)

if ($Ret.ReturnValue -gt 0)
```

```
{
   write-Host "Flush time interval NOT changed on disk" $Disk "- Error:"
$Ret.ReturnValue
}
else
{
   write-Host "Flush time interval set to" $TimeInt "on disk" $Disk "on next boot"
}


}
else
{
write-Host "HPDC_SetFlushTimeInt.ps1 needs disk number and flush time interval"
write-Host "For Example:  PowerShell .\HPDC_SetFlushTimeInt.ps1 0 60"
}
```

SetFlushCount

Sets the cache allocation policy for a specified disk

Syntax

```
[Static] uint32 SetFlushCount
([IN] uint32 DiskNumber),
([IN] uint32 FlushCount);
```

Return Value

Returns 0 when successful. Otherwise, it returns an error code.

PowerShell Script

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$Disk=$args[0]
if (!$Disk)
{
  $Disk=0
}
$Count=$args[1]

if ($Count)
{

$HpDC = [WmiClass]"root\hp\writefilters:HpCacheDiskConfig"
$Ret=$HpDC.SetFlushCount($Disk,$Count)

if ($Ret.ReturnValue -gt 0)
{
   write-Host "Flush count NOT changed on disk" $Disk "- Error:" $Ret.ReturnValue
```

```
}
else
{
   write-Host "Flush count set to" $Count "on disk" $Disk "on next boot"
}

}
else
{
write-Host "HPDC_SetFlushCount.ps1 needs disk number and flush count"
write-Host "For Example:  PowerShell .\HPDC_SetFlushCount.ps1 0 10"
}
```

Cache statistics for received IO

HpCacheSystemStatistics contains statistics on IO received by the HP Cache driver from the operating system.

Syntax and Property Descriptions

```
class HpCacheSystemStatistics{

[Key,Read] string DiskId;
[Key,Read] uint32 DiskNumber;

PropertyContext("TotalWritesServiced"): ToInstance,
  [Dynamic, Provider("HpCacheProvider")]
       sint32 TotalWritesServiced;

PropertyContext("ReadsServiced"): ToInstance,
  [Dynamic, Provider("HpCacheProvider")]
       sint32 ReadsServiced;

PropertyContext("ReadsServicedFromCache"): ToInstance,
  [Dynamic, Provider("HpCacheProvider")]
       sint32 ReadsServicedFromCache;

PropertyContext("TotalBytesRead"): ToInstance,
  [Dynamic, Provider("HpCacheProvider")]
       sint64 TotalBytesRead;

PropertyContext("TotalFlushesRequested"): ToInstance,
```

```
  [Dynamic, Provider("HpCacheProvider")]
       sint32 TotalFlushesRequested

}
```

Table 10. Property descriptions for HpCacheSystemStatistics

Property	Data type	Qualifiers	Description
DiskID	string	[Key, Read]	The persistent Disk ID of the disk.
DiskNumber	uint32	[Key, Read]	The disk number of the disk.
TotalWritesServiced	sint32	[Read]	The total number of write operations sent by the operating system to the specified disk.
TotalBytesWritten	sint32	[Read]	The total number of bytes sent by the operating system to the specified disk.
ReadsServiced	sint32	[Read]	The number of read operations sent by the operating system.
ReadsServicedFromCache	sint32	[Read]	The number of read operations sent by the operating system that were serviced by the cache, rather than by the disk.
TotalBytesRead	sint32	[Read]	The total number of bytes read by the operating system from the specified disk.
TotalFlushesRequested	sint32	[Read]	The total number of flushes requested by the system for this disk.

PowerShell Script for HPCacheSystemStatistics

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$Props = Get-WmiObject -namespace root\hp\writefilters HpCacheSystemStatistics

write-Host " "
write-Host "Statistics on IO received by Cache drive from OS"

foreach ($x in $Props) {

        write-Host " "
  write-Host "Disk ID: " $x.DiskId
  write-Host "Disk Number : " $x.DiskNumber
  write-Host "Total number of writes sent by OS to disk: " $x.TotalWritesServiced
```

```
  write-Host "Total number of bytes sent by OS to disk: " $x.TotalBytesWritten
  write-Host "Number of reads sent by OS: " $x.ReadsServiced
  write-Host "Number of reads sent by OS serviced by cache instead of disk: "
$x.ReadsServicedFromCache
  write-Host "Total number of bytes read by OS from disk: " $x.TotalBytesRead
  write-Host "Total number of flushes request by system for disk: "
$x.TotalFlushesRequested

}
```

Cache statistics for IO no longer in cache

HpCacheCacheStatistics contains statistics on IO that has left the cache.

Syntax and Property Descriptions

```
class HpCacheCacheStatistics{

[Key,Read] string DiskId;
[Key,Read] uint32 DiskNumber;

PropertyContext("TotalFlushes"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  sint32 TotalFlushes;

PropertyContext("FlushWithPurge"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  sint32 FlushWithPurge;

PropertyContext("FlushesDueToCacheFull"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  sint32 FlushesDueToCacheFull;

PropertyContext("FlushesDueToCounter"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  sint32 FlushesDueToCounter;

PropertyContext("FlushesDuetoTimer"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  sint32 FlushesDueToTimer;

PropertyContext("TotalWritesPerformed"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
```

```
  sint32 TotalWritesPerfromed;

PropertyContext("TotalBytesWritten"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  sint64 TotalBytesWritten;
}
```

Table 11. Property descriptions for HpCacheCacheStatistics

Property	Data type	Qualifiers	Description
DiskID	string	[Key, Read]	The persistent Disk ID of the disk.
DiskNumber	uint32	[Key, Read]	The disk number of the disk.
TotalFlushes	sint32	[Read]	The total number cache flushes that have occurred since caching began.
FlushwithPurge	sint32	[Read]	The total number cache flushes with purges that have occured since caching began.
FlushesDueToCacheFull	sint32	[Read]	The total number cache flushes that have occurred due to a low cache space condition.
FlushesDuetoCounter	sint32	[Read]	The total number cache flushes that have occurred due to the FlushCount being reached.
FlushesDueToTimer	sint32	[Read]	The total number cache flushes that have occurred due to the cache timer expiring.
TotalWritesPerformed	sint32	[Read]	The total number of write operations sent to the disk from the cache.
TotalBytesWritten	sint64	[Read]	The total number of bytes written to the disk from the cache.

PowerShell Script for HPCacheCacheStatistics

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$props = Get-WmiObject -namespace root\hp\writefilters HpCacheCacheStatistics

write-Host " "
write-Host "Statistics on IO that has left the cache"

foreach ($x in $props) {

        write-Host " "
```

```
  write-Host "Disk ID: " $x.DiskId

  write-Host "Disk Number : " $x.DiskNumber

  write-Host "Total cache flushes since caching began: " $x.TotalFlushes

  write-Host "Total cache flushes with purges since caching began: "
$x.FlushWithPurge

  write-Host "Total cache flushes due to low space: " $x.FlushesDueToCacheFull

  write-Host "Total cache flushes due to flush count being reached: "
$x.FlushesDueToCounter

  write-Host "Total cache flushes due to cache timer expiring: "
$x.FlushesDueToTimer

  write-Host "Total number of writes sent to disk from cache: "
$x.TotalWritesPerformed

  write-Host "Total number of bytes written to disk from cache: "
$x.TotalBytesWritten


}
```

Cache behavior information

The HpCacheDiskInfo reports HP Cache's behavior for a given disk

Syntax and Property Descriptions

.

```
class HpCacheDiskInfo {


[Key,Read] string DiskId;

[Key,Read] uint32 DiskNumber;


PropertyContext("CachingEnabled"): ToInstance,

 [Dynamic,Provider("HpCacheProvider")]

 boolean CachingEnabled;


PropertyContext("AllocationPolicy"): ToInstance,

 [Dynamic,Provider("HpCacheProvider")]

  string AllocationPolicy;


PropertyContext("MaximumCacheSize"): ToInstance,

 [Dynamic,Provider("HpCacheProvider")]

  sint64 MaximumCacheSize;
```

```
PropertyContext("CurrentCacheUsage"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  sint64 CurrentCacheUsage;

PropertyContext("CurrentCacheAllocation"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  sint64 CurrentCacheAllocation;

PropertyContext("FlushTimeInterval"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  uint32 FlushTimeInterval;

PropertyContext("FlushCount"): ToInstance,
 [Dynamic,Provider("HpCacheProvider")]
  uint32 FlushCount;
```

Table 11. Property descriptions for HpCacheDiskInfo

Property	Data type	Qualifiers	Description
DiskID	string	[Key, Read]	The persistent Disk ID of the disk.
DiskNumber	uint32	[Key, Read]	The disk number of the disk.
CachingEnabled	boolean	[Read]	The total number cache flushes that have occurred since caching began.
AllocationPolicy	string	[Read]	Gets the cache allocation policy for this disk.
MaximumCacheSize	sint64	[Read]	The maximum size that the cache can grow to. If the AllocationPolicy is 'Static', the cache will be instantiated at this size.
CurrentCacheUsage	sint64	[Read]	The current size of the cache, in bytes.
CurrentCacheAllocation	sint64	[Read]	The current allocated size of the cache, in bytes.
FlushTimeInterval	sint32	[Read]	The flush interval of the cache. Data will not remain in the cache for longer than this interval.
FlushCount	uint32	[Read]	The number of system flush operations seen before causing a flush of the HpCache cache.

PowerShell Script for HPCacheDiskInfo

The following text is the PowerShell script code. Copy and paste this text and save as a .ps1file.

```
$Props = Get-WmiObject -namespace root\hp\writefilters HpCacheDiskInfo

write-Host " "
write-Host "Cache behavior for each disk"

foreach ($x in $Props) {

        write-Host " "
  write-Host "Disk ID: " $x.DiskId
  write-Host "Disk Number : " $x.DiskNumber
  write-Host "Is caching enabled?: " $x.CachingEnabled
  write-Host "Allocation policy is: " $x.AllocationPolicy
  write-Host "Maximum size of cache (in bytes): " $x.MaximumCacheSize
  write-Host "Current size of the cache (in bytes): " $x.CurrentCacheUsage
        write-Host "Current allocated size of the cache (in bytes): "
$x.CurrentCacheAllocation
  write-Host "Flush interval of the cache (in seconds): " $x.FlushTimeInterval
  write-Host "Number of flushes before write to disk: " $x.FlushCount
}
```

For more information

Go to this link to see more information on using .mof files

https://technet.microsoft.com/en-us/library/cc180827.aspx

Go to this link to see more information on WMI scripting

https://technet.microsoft.com/en-us/library/bb684733.aspx

Sign up for updates
hp.com/go/getupdated

First Edition: October 2018

Document Part Number: L42037-001