Microsoft® Forefront™

Forefront Management Shell

PowerShell Management of Forefront Server Products

Published: October, 2009

Software version: Forefront Protection 2010 for Exchange Server

Mitchell Hall

Microsoft®

Contents

Introduction

PowerShell is becoming the standard command line interface for Microsoft's server products. This standardization has many benefits:

- PowerShell provides a consistent user interface with a large set of built-in commands. These built-in commands provide useful features and are common in all PowerShell implementations. This provides product-to-product consistency.
- PowerShell command design is covered by rigid syntax guidelines. PowerShell commands are called Cmdlets. These Cmdlets are made up of two parts: verbs (the actions) and nouns (the areas of the product). There are a limited number of defined verbs, with specific definitions. This provides Cmdlet consistency across the products.
- PowerShell interfaces are called snap-ins. Each snap-in uses the same command parser, thus the command processing is consistent.
- PowerShell parameters can be defined with range and value validations.
- The PowerShell constructs offer an Enum (Enumeration) type. This type provides distinct value checking, thus reducing the likelihood of an error.
- PowerShell itself does not process text. It is a .NET based interface that processes objects. This helps to provide a more powerful scripting interface than standard command lines interfaces.
- Scripting in the PowerShell environments provides access to powerful scripting functionality by using the built in libraries.

PowerShell usage overview

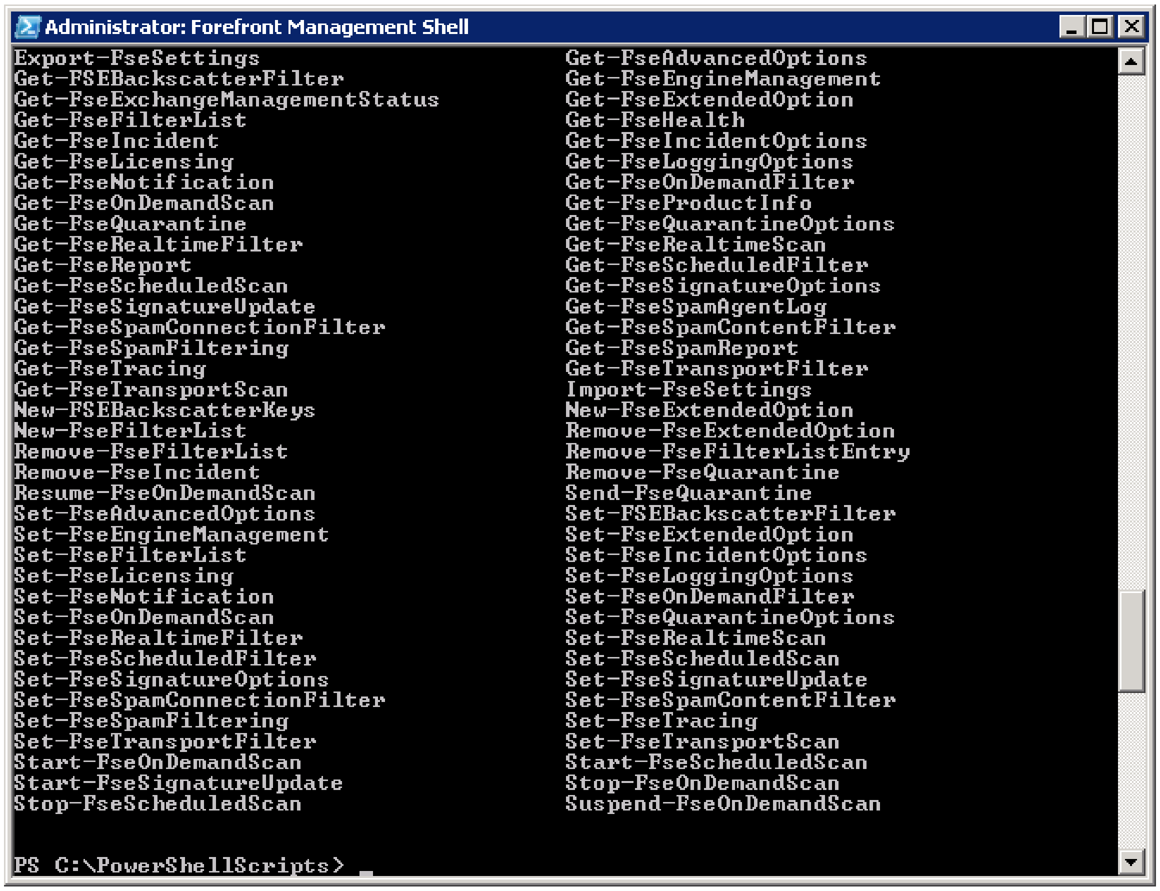
The Forefront Management Shell provides a fully scriptable interface into Forefront Protection 2010 for Exchange Server (FPE). The FPE administrator console is implemented on top of the PowerShell interface, providing assurance that all functionality provided in the console is also implemented within PowerShell.

In order to use the Forefront Management Shell effectively, you should first familiarize yourself with PowerShell. When you open the Forefront Management Shell, the system loads a copy of the Forefront Management snap-in, inside a PowerShell instance. To gain access to the Forefront cmdlets, select Forefront Management Shell from the Microsoft Forefront Server Security program group. Using the "Windows PowerShell" progam link will not load the Forefront snap-in, and the Forefront Cmdlets will not be available.

Once the Forefront Management Shell is loaded, the list of Forefront cmdlets can be displayed by using the command:

```
Get-Command *-fs* | fw name
```

Example:


Administrator: Forefront Management Shell

```
Export-FseSettings                      Get-FseAdvancedOptions
Get-FSEBackscatterFilter                Get-FseEngineManagement
Get-FseExchangeManagementStatus         Get-FseExtendedOption
Get-FseFilterList                       Get-FseHealth
Get-FseIncident                         Get-FseIncidentOptions
Get-FseLicensing                        Get-FseLoggingOptions
Get-FseNotification                     Get-FseOnDemandFilter
Get-FseOnDemandScan                     Get-FseProductInfo
Get-FseQuarantine                       Get-FseQuarantineOptions
Get-FseRealtimeFilter                   Get-FseRealtimeScan
Get-FseReport                           Get-FseScheduledFilter
Get-FseScheduledScan                    Get-FseSignatureOptions
Get-FseSignatureUpdate                  Get-FseSpamAgentLog
Get-FseSpamConnectionFilter             Get-FseSpamContentFilter
Get-FseSpamFiltering                    Get-FseSpamReport
Get-FseTracing                          Get-FseTransportFilter
Get-FseTransportScan                    Import-FseSettings
New-FSEBackscatterKeys                  New-FseExtendedOption
New-FseFilterList                       Remove-FseExtendedOption
Remove-FseFilterList                    Remove-FseFilterListEntry
Remove-FseIncident                      Remove-FseQuarantine
Resume-FseOnDemandScan                  Send-FseQuarantine
Set-FseAdvancedOptions                  Set-FSEBackscatterFilter
Set-FseEngineManagement                 Set-FseExtendedOption
Set-FseFilterList                       Set-FseIncidentOptions
Set-FseLicensing                        Set-FseLoggingOptions
Set-FseNotification                     Set-FseOnDemandFilter
Set-FseOnDemandScan                     Set-FseQuarantineOptions
Set-FseRealtimeFilter                   Set-FseRealtimeScan
Set-FseScheduledFilter                  Set-FseScheduledScan
Set-FseSignatureOptions                 Set-FseSignatureUpdate
Set-FseSpamConnectionFilter             Set-FseSpamContentFilter
Set-FseSpamFiltering                    Set-FseTracing
Set-FseTransportFilter                  Set-FseTransportScan
Start-FseOnDemandScan                   Start-FseScheduledScan
Start-FseSignatureUpdate                Stop-FseOnDemandScan
Stop-FseScheduledScan                   Suspend-FseOnDemandScan


PS C:\PowerShellScripts> _
```

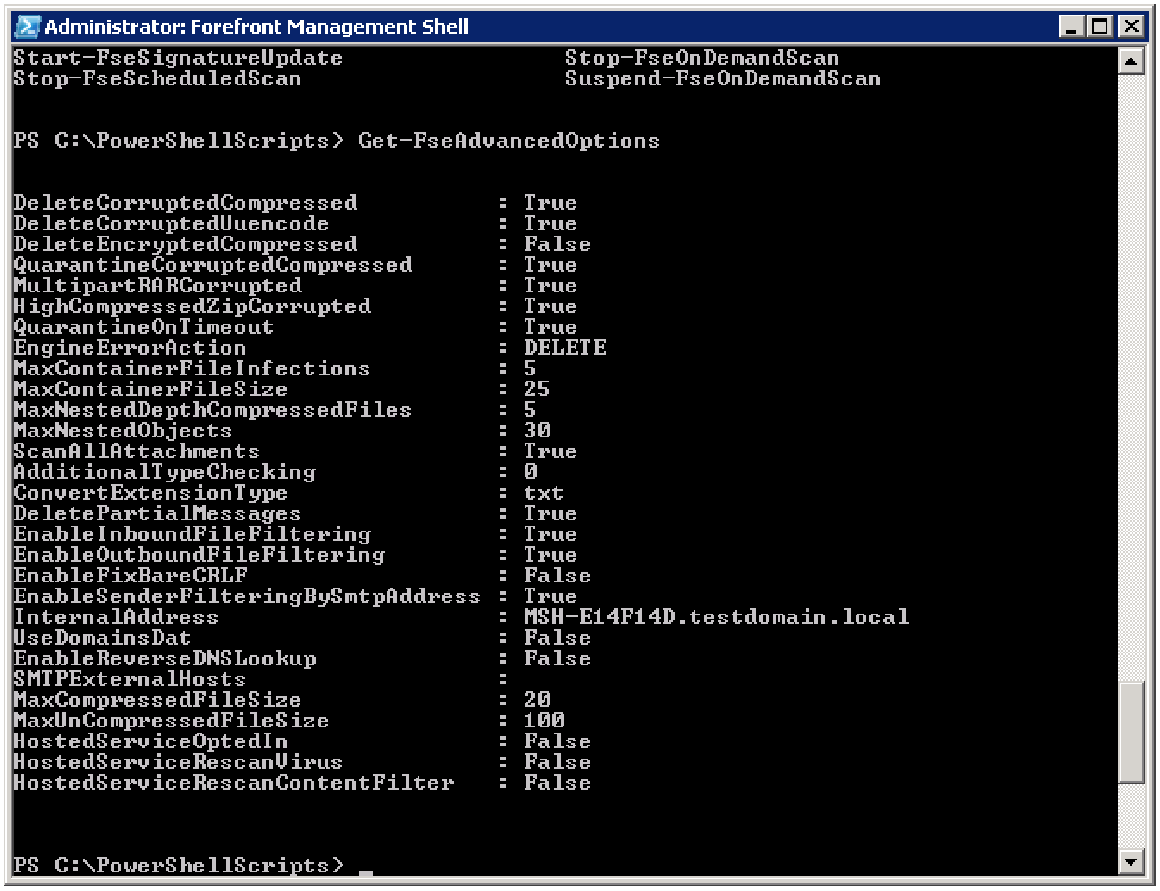
This example illustrates a few things about the PowerShell interface.

1. The Get-Command PowerShell command is used to retrieve commands based on the filter provided. In this case “*-fs*”
2. PowerShell provides a pipeline feature that allows the output of one command to be the input to the next command. In this case the output of Get-Command *-fs* is piped, by use of the “|” character, to be the input of the PowerShell “fw” (format wide) cmdlet. There are also other built-in formatting cmdlets such as ft (format table) and fl (format list). These are native PowerShell commands that can be used to format the standard output of PowerShell cmdlets.
3. The fs cmdlet takes the output of a cmdlet, in this case Get-Command, and displays the subset of values. In this case the subset is “name”.

To obtain help on an individual PowerShell CmdLet you can use the built in Get-Help CmdLet. The syntax is:

Get-Help *cmdlet_name*

where *cmdlet_name* is the cmdlet for which you want to see help.



```
Start-FseSignatureUpdate                Stop-FseOnDemandScan
Stop-FseScheduledScan                   Suspend-FseOnDemandScan

PS C:\PowerShellScripts> Get-FseAdvancedOptions


DeleteCorruptedCompressed          : True
DeleteCorruptedUuencode            : True
DeleteEncryptedCompressed          : False
QuarantineCorruptedCompressed      : True
MultipartRARCorrupted              : True
HighCompressedZipCorrupted         : True
QuarantineOnTimeout                : True
EngineErrorAction                  : DELETE
MaxContainerFileInfections         : 5
MaxContainerFileSize               : 25
MaxNestedDepthCompressedFiles      : 5
MaxNestedObjects                   : 30
ScanAllAttachments                 : True
AdditionalTypeChecking             : 0
ConvertExtensionType               : txt
DeletePartialMessages              : True
EnableInboundFileFiltering         : True
EnableOutboundFileFiltering        : True
EnableFixBareCRLF                  : False
EnableSenderFilteringBySmtpAddress : True
InternalAddress                    : MSH-E14F14D.testdomain.local
UseDomainsDat                      : False
EnableReverseDNSLookup             : False
SMTPExternalHosts                  :
MaxCompressedFileSize              : 20
MaxUnCompressedFileSize            : 100
HostedServiceOptedIn               : False
HostedServiceRescanVirus           : False
HostedServiceRescanContentFilter   : False



PS C:\PowerShellScripts> _
```

Forefront PowerShell functional areas

Forefront functionality has the following logical groupings: Engine updates, system/advanced options, import/export, incidents/quarantine, notifications, reporting, scanning, filter lists, and spam filtering. Within these groupings we define the sets of commands for the functionality. By combining the groupings (nouns) with clearly defined actions (verbs), we provide a compact set of commands grouped with common functionality.

Engine Updates Cmdlets

This functional area includes managing the engine update schedule, engine selections for each scan job, and proxy server configuration commands.

Cmdlet	Description
Get-FseSignatureUpdate	Retrieves the engine update schedule
Set-FseSignatureUpdate	Sets the engine update schedule
Start-FseSignatureUpdate	Initiates an engine update
Get-FseSignatureOptions	Retrieves settings for connection and proxy configuration
Set-FseSignatureOptions	Sets connection and proxy configuration parameters

System and Advanced Options Cmdlets

This functional area controls system processing, including logging and tracing options, handling of encrypted files, and container navigation.

Cmdlet	Description
Get-FseAdvancedOptions	Sets various scan options that are not part of other cmdlets. These include compressed file handling, the action to take when an engine error occurs, additional type checking, enabling in-bound and outbound file filtering, specifying internal addresses, specifying external hosts, and enabling reverse Domain Name System (DNS) look-ups
Set-FseAdvancedOptions	Sets various scan options that are not part of other cmdlets. These include compressed file handling, the action to take when an engine error occurs, additional type checking, enabling in-bound and outbound file filtering, specifying internal addresses, specifying external hosts, and enabling reverse Domain Name System (DNS) look-ups.
Get-FseLoggingOptions	Retrieves the logging level for various areas of the product. Log-ging includes transport logging, incident logging, performance logging, and event logging.
Set-FseLoggingOptions	Enables logging for individual areas of the product.
Get-FseTracing	Retrieves system tracing information, such as log size, flush fre-quency, and verbosity level.
Set-FseTracing	Sets trace settings. Adjustments to these settings are usually

	made with the guidance of support personnel.

Import and Export Settings

This functional area provides the ability to back up and restore the system configuration.

Cmdlet	Description
Import-FseSettings	Imports the settings from a previously exported XML file. This functionality is provided as a way to back up and restore a complete configuration
Export-FseSettings	Exports all configuration settings. This is to support backing up and restoring full configurations

Incidents and Quarantine

This functional area allows you to manage incidents, which are records of different detection types, as well as quarantine, which contains mail records withheld from the mail stream due to malware detection or filter match.

Cmdlet	Description
Get-FseIncidentOptions	Retrieves the incidents database options
Set-FseIncidentOptions	Sets the incident database options. This includes settings such as size, purging, and retention
Get-FseIncident	Retrieves an individual incident from the database
Remove-FseIncident	Retrieves the settings used to direct quarantine processing
Get-FseQuarantineOptions	Retrieves the settings used to direct quarantine processing
Set-FseQuarantineOptions	Sets the quarantine options, including purging and retention settings
Get-FseQuarantine	Retrieves quarantine metadata from the database. The data can be filtered to retrieve a subset of the complete data
Remove-FseQuarantine	Removes items from the quarantine database. Allows the selection of individual items or all items
Export-FseQuarantine	Exports quarantine data to disk. This command allows potentially dangerous content to be moved out of quarantine
Send-FseQuarantine	Delivers quarantined mail to the original recipients or to additional recipients

Notifications

This functional area provides access to the notifications that can be configured for the system. Notifications are e-mail messages that can be sent to administrators and others when certain events occur. Administrators can be alerted when viruses are found and when other critical events are triggered.

Cmdlet	Description
Get-FseNotification	Retrieves the settings for all of the e-mail notifications
Set-FseNotification	Sets the e-mail notification options. You can configure individual e-mail notifications for each event, such as virus detection

Reporting

Forefront maintains a series of counters for different detection types. They are available to the user through a series of PowerShell commands.

Cmdlet	Description
Get-FseReport	Retrieves reports that show the activity for malware detections, file filter matches, keyword filter matches, sender-domain filter matches, subject line filter matches, all filter matches, and all detections
Clear-FseReport	Resets the statistics for the report type specified
Get-FseSpamReport	Retrieves a report detailing the individual spam counters

Scanning

Scanning is the process that the system uses to identify malicious content and spam. It is also the same process that is used to filter mail by comparing it to the filter lists that an administrator can create. Each type of scanning is done by a scan job with its own settings.

Cmdlet	Description
Get-FseTransportScan	Retrieves settings for the transport scan
Set-FseTransportScan	Applies settings for the transport scan. Settings include enabled, filtering types to apply, directional scanning, scanning types (Spyware, antivirus, filtering), and engine selection
Get-FseRealtimeScan	Retrieves settings for the realtime scan

Set-FseRealtimeScan	Applies settings for the realtime scan. Settings include enabled, scanning types (Spyware, antivirus, filtering), mailbox list, public folder list, engine selection, and process count
Get-FseScheduledScan	Retrieves settings for the scheduled scan
Set-FseScheduledScan	Applies settings for the scheduled scan job. Settings include enabled, frequency, window of scanning, scanning types (Spy-ware, antivirus, filtering), mailbox list, public folder list, engine selection, and process count
Start-FseScheduledScan	Initiates a scheduled scan
Stop-FseScheduledScan	Stops a scheduled scan
Get-FseOnDemandScan	Retrieves the settings for the on-demand scan
Set-FseOndemandScan	Applies settings to the on-demand scan. Settings include a mail-box list, public folder list, engine selection, and deletion text.
Start-FseOndemandScan	Initiates an on-demand scan. The primary purpose of this com-mand is to immediately scan a small number of mailboxes or public folders
Stop-FseOndemandScan	Stops a currently active on-demand scan
Suspend-FseOndemandScan	Pauses an on-demand scan.
Resume-FseOndemandScan	Restarts a suspended on-demand scan

Filter Lists

Filtering means matching the contents of an e-mail to a set of predefined lists of words or patterns. These words and patterns are defined in filter lists. These filter lists are then associated with the scan jobs defined above to provide the filtering functionality. In addition to filtering on words and patterns, the system also has the ability to filter by file type, file name, file extension, file size, and several other criteria. An example of a use of filter lists is provided in the section "**Example of a filtering script**" below.

There are three basic steps to using filter lists:

1. Create an empty filter list by using the New-FseFilterlist cmdlet.
2. Add entries to the filter list by using the Add-FseFilterListEntry cmdlet.
3. Bind a filter list to a scan job by using one of the Set commands, for example: Set-FseRealtimeFilter.

Cmdlets to create and modify filter lists

Cmdlet	Description
New-FseFilterList	Creates a new filter list. The list can be empty (to be filled in later with Add-FseFilterListEntry) or you can populate it when you create it. The list has a name and ~~a~~ you must indicate the filter list type. The types include file filters(-File), subject filters (-Subject), sender-domain filters(-SenderDomain), keyword fil-ters(-Keyword), and allowed sender filters(-AllowedSender)
Get-FseFilterList	Retrieves an individual filter list
Set-FseFilterList	Replaces all the values of an existing filter list
Clear-FseFilterList	Clears the contents of an existing filter list
Remove-FseFilterList	Deletes an existing filter list

Cmdlets to add and remove entries from filter lists

Cmdlet	Description
Add-FseFilterListEntry	Adds one or more entries to an existing filter list
Remove-FseFilterListEntry	Removes one or more items from a filter list

Cmdlets that bind filter lists to scan jobs and retrieve settings

Cmdlet	Description
Get-FseScheduledFilter	Retrieves the configuration of all filter lists of a particular type that were enabled for the scheduled scan
Set-FseScheduledFilter	Configures a filter list and associates it with the scheduled scan. Settings include action on detection and quarantine options
Get-FseOnDemandFilter	Retrieves the configuration of all filter lists of a particular type that were enabled for the on-demand scan
Set-FseOnDemandFilter	Configures a filter list and associates it with the on-demand scan. Settings include action on detection and quarantine options
Get-FseTransportFilter	Retrieves the configuration of all filter lists of a particular type that were enabled for the transport scan
Set-FseTransportFilter	Configures a filter list and associates it with the transport scan. Settings include action on detection and quarantine options

Get-FseRealtimeFilter	Retrieves the configuration of all filter lists of a particular type that were enabled for the realtime scan
Set-FseRealtimeFilter	Configures a filter list and associates it with the realtime scan. Settings include action on detection and quarantine options

Spam Filtering

Cmdlet	Description
Get-FseSpamConnectionFilter	Retrieves configuration settings for the spam connection filter (DNS Block List)
Set-FseSpamConnectionFilter	Sets configuration options for the Forefront DNS Block List (DNSBL)
Get-FseSpamContentFilter	Retrieves the settings for the spam content filter
Set-FseSpamContentFilter	Sets the configuration options for the spam content filter. This includes options for allowed recipients, allowed domains, action on detection, quarantine, and Spam Confidence Level threshold
Get-FseSpamFiltering	Retrieves the setting for the spam filtering
Set-FseSpamFiltering	Applies the enable setting for spam filtering
Get-FseSpamAgentLog	Retrieves log records for all Forefront spam agents to aid in troubleshooting. This command is a script not a cmdlet. It is located in the program directory where Forefront was installed. The purpose of this script is to allow administrators to pull log information from various agent logs

Example of a filtering script

In Notepad create a new file with the following lines:

```
New-FseFilterList -List MySubjectList –Subject

Add-FseFilterListEntry -Subject -list MySubjectList hello, goodbye, resume, ssn

Set-FseOnDemandFilter -Subject -List MySubjectList -Enabled $true -Action SkipDetect
```

Save the file to the name Example.ps1.

To execute the command, open the Forefront Management Shell from the Start menu. Navigate to the directory that contains the file you just created. At the command prompt type:

```
.\Example.ps1
```

Now let’s take a look at each line to determine what it does.

Line 1: New-FseFilterList -List MySubjectList –Subject

Create a new subject filter list named "MySubjectList"

Line 2: Add-FseFilterListEntry -Subject -list MySubjectList hello, goodbye, resume, ssn

Add a number of entries to "MySubjectList". These are the items that the filter will attempt to match.

Line 3: Set-FseOnDemandFilter -Subject -List MySubjectList -Enabled $true -Action SkipDetect

Binds "MySubjectList" to the on-demand scan, enables it, and sets its action to record any matches, but to let the e-mail go through (SkipDetect).

Now when the OnDemandScan is run, it will include the "MySubjectList" subject filter, along with the other scanning it does.

Forefront PowerShell automation

Scripting in the PowerShell environment exposes the full .NET Framework suite of functionality. This can be used in conjunction with the Forefront PowerShell objects to provide a complete programming environment.

The following example is a script that will examine the update status of each of the engines and print a list of the engines that have not been updated since the date provided. This script can be created in Notepad.

```
function Pause ($Message="More....`n")
{
Write-Host  $Message
$bucket = $Host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown")

}


#retrieve the date augement from the command line
$CheckDate = $Args[0]

if ($CheckDate -eq $null)
{
write-host -Foregroundcolor red "`nPlease enter a date to check for.`n`n"
}
else
{
  echo("`nGet a list of the Engines and their status information")
  echo("`nGet-FSESignatureUpdate")

  Get-FSESignatureUpdate| ft Engine, LastCheck

  $B = Get-FSESignatureUpdate
   pause
  echo("`nLoop through list looking for outdated engines")

# loop through each engine and see if the LastCheck date is less than the date
# provided in the command line
Foreach( $c in $b)
{
    if ($c.LastCheck -lt $CheckDate)
  {
  $d = [string]$c.Engine + " - " + [string]$c.LastCheck

  write-host -Foregroundcolor red $d
  }

}

write-host "`n`n`n`n"

}
```

Save the file to EngineCheck.ps1.

Open for Forefront Management Shell, navigate to the directory where you stored the script and execute the command:

```
.\EngineCheck.ps1 <todays date>
```

For example

```
.\EngineCheck.ps1 12/31/2008
```

Multiple Server Management via PowerShell

With the functionality of the FSE PowerShell interface it is possible for you to automate the deployment of settings to multiple machines. Typical settings can be exported by using the Export-FseSettings command. In addition to the settings, there are advanced settings that are not contained as part of the settings export. These settings should only be modified when instructed by Microsoft support personnel. In order to automate the export and import of these settings, a script can be written to execute the export commands directing output to text files. These text files can be moved to another server or placed on a network share. These text file can be then be read by another PowerShell script and execute appropriate commands on the target server. Below is an example of scripts that can be used to save/restore settings and extended options.

Example of Script to save both settings and extended options

```
param ([string]$Path="")

if ($path -eq "")
{
  $path = Get-Location -PSProvider "FileSystem"
}
```

```
$cmd = "Export-FseSettings -Path " + $path + "\ExportSettings.txt"
$cmd
Invoke-Expression ($cmd)

$cmd = "get-FseExtendedOption -Name *  | fl name,value > " + $path  +
"\ExtendedOptions.txt"
$cmd
Invoke-Expression ($cmd)
```

Example of Script to restore both settings and extended options

```
param ([string]$path="")

if ($path -eq "")
{
  $path = Get-Location -PSProvider "FileSystem"
}

$filename=$path + "\ExportSettings.txt"
$cmd = "Import-FseSettings -Path " + $filename
$cmd
Invoke-Expression ($cmd)

$filename=$path + "\ExtendedOptions.txt"

$Options = get-content $filename

    foreach ($Option in $Options)
    {
       $Test = $Option.Trim()
       if ($Test.Length -gt 4)
       {
          if ($Test.ToUpper().SubString(0,4) -eq "NAME")
          {
          $name = $Test.ToUpper().SubString(6,$Test.Length-6)
          $name = $Test.SubString($Test.Indexof(":")+1, $Test.Length-$Test.Indexof(":")-
1).Trim()
          }
          if ($Test.ToUpper().SubString(0,5) -eq "VALUE")
          {
          $value = $Test.ToUpper().SubString(6,$Test.Length-6)
          $value = $Test.SubString($Test.Indexof(":")+1, $Test.Length-$Test.Indexof(":")-
1).Trim()
          If ($cmd -ne ""-and $value -ne "")
            {
              $cmd = "Set-FseExtendedOption -Name " + $name + " -Value " + $Value
              $cmd
              Invoke-Expression ($cmd)
            }
          }
       }

     }
```

Summary

The PowerShell interface to Forefront Protection 2010 for Exchange Server provides a powerful interface to allow the management of all aspects of the product. This interface provides access to the following features:

- The Forefront Management Shell provides a fully functional PowerShell interface that enables administrators to manage the product.
- PowerShell cmdlets are comprised of a Verb – Noun pair. The Forefront PowerShell commands are logically grouped according to functionality (verbs). The actions (verbs) associated with the functionality are based on a well defined set of actions. This provides a well defined grouping of PowerShell commands, which aids in discoverability and usability.
- A PowerShell interface provides a method for administrators to automate common functionality. Administrators can now automate common tasks through existing management tools.
- PowerShell scripting provides a powerful programming environment for working with Forefront objects. The PowerShell scripting language provides the full .Net object support.