



Enterprise Vault Whitepaper

Advanced Strategies for Monitoring Enterprise Vault

This document is about developing a strategy to monitor the Enterprise Vault environment using native tools such as Windows operating system utilities and Microsoft SQL queries.

If you have any feedback or questions about this document please email them to EV-TFE-Feedback@symantec.com stating the document title. .

This document applies to the following version(s) of Enterprise Vault:
8.0, 9.0, 10.0

This document is provided for informational purposes only. All warranties relating to the information in this document, either express or implied, are disclaimed to the maximum extent allowed by law. The information in this document is subject to change without notice. Copyright © 2011 Symantec Corporation. All rights reserved. Symantec, the Symantec logo and Enterprise Vault are trademarks or registered trademarks of Symantec Corporation or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners

Document Control

Contributors

Who	Contribution
Patti Rodgers Symantec Business Critical Services	Author
Rob Wilcox, Benoit Lionnet, Chris Warren, and Wolfgang Rebhan	Contributions and review

Revision History

Version	Date	Changes
1.0	March 2012	Initial release

Related Documents

Version	Date	Title

Table of Contents

Introduction and Methodology	1
Background	1
Deciding What and How to Monitor	1
Applying an Action	2
Who to Notify	2
Know the Environment	2
Tracking Expected Activities	3
About Enterprise Vault Operations Manager	4
Approaches	5
Reactive	5
Proactive	6
Establishing a Baseline	6
Data Repositories and Tools	8
Windows Event Logs	8
Setting Up Notifications from Event Logs	9
Managing Multiple Event Logs	12
Windows Performance Monitor (PerfMon)	17
Working With Performance Monitor Statistics	17
Configuring Performance Monitor Alerts	18
Scheduling Performance Monitor Using Logman	21
SQL Databases	23
Mining SQL Data	23
Using SQL Jobs	23
IIS Logs	26
Mining the Enterprise IIS Logs	27
Microsoft Message Queuing (MSMQ)	28
Using the Typeperf Command to View MSMQ Statistics	29
Client Registry	29
Automating Registry Export and Collection	30
Tapping into DTrace	30

Appendices

APPENDIX - Sample Scripts, Batchfiles and Syntax

Introduction and Methodology

Background

As the Enterprise Vault environment grows, user reliance on archived data also grows; ensuring users have reliable access to archived data becomes an Enterprise Vault Administrator's highest priority. To that end, customers must develop an efficient and effective monitoring strategy to help ensure Administrators have needed system information with as little effort as possible. This document will explore some approaches that the Enterprise Vault customer can use to develop a custom monitoring strategy.

This document assumes the customer has intermediate level experience with Symantec Enterprise Vault, the Microsoft Windows operating systems, Microsoft SQL Server, and Microsoft Exchange server, as well as basic scripting skills. The available tools and syntax will vary with the operating system and Microsoft SQL version; consult relevant product documentation for version-specific instructions.

There is no single, comprehensive monitoring strategy that would be relevant to all customers; each customer must evaluate their own business goals and standards.. This document will introduce customers to many of the tools and techniques to gather relevant information. These techniques can be used in conjunction with any of the third-party monitoring suites, or on their own as stand-alone routines.

Deciding What and How to Monitor

One of the biggest obstacles in creating an effective monitoring strategy is deciding what to monitor. The monitored conditions must be both granular and relevant. It is important to strike the proper balance between monitoring too much and monitoring too little; by monitoring too little, important conditions or events may be missed, leading to an increased problem response time, and by monitoring too much, Administrators become desensitized to alerts and warnings, which dilutes the overall effectiveness of the monitoring strategy. The end goal should be to monitor only those conditions that are important, keeping the list of monitored criteria as small as possible while still providing the information the Administrator needs. Administrators should not bury themselves in hundreds of daily alert emails.

As an example, an event is written to the Windows Event Logs when Enterprise Vault is unable to archive from a user's mailbox because the user is over their Exchange quota; this condition will be important in some environments and unimportant in others. By determining if the event is important, the Administrator can decide whether or not to monitor for it. Generally speaking, if there is Administrator intervention available or required, then the event is more likely to be considered important. If normal procedures would not include contacting the user to assist in reducing Exchange mailbox size, then the event would probably not be considered important enough to monitor.

Each Administrator must review the available options, determine which are relevant, and focus on the events and conditions for which some kind of corrective action can be taken. The monitoring strategy

should be reviewed periodically and adjusted as needed, as business needs are constantly evolving; a change to the use or importance of the Enterprise Vault environment can change the Administrator's approach to monitoring.

Applying an Action

Associate an action with the monitoring routine. The action can be a simple notification (i.e. send an email to the Administrator or write an event to the Event Logs) or can apply a corrective action (call a batchfile to fix a specific problem). If a condition has a known, consistent and reproducible solution, automating the corrective action can help ensure maximum uptime with as little administrative overhead as possible; however, take care that some notation or logging is made when the corrective action is applied, so that Administrators may be able to see patterns and apply permanent solutions. In other words, while restarting Archiving tasks that fail due to certain MAPI errors helps ensure uptime, an Administrator should also be able to identify a systemic problem. A long-term solution that prevents the problem is better than a recurring automated solution.

Who to Notify

Most notifications can be delivered by email; selecting a distribution list as the recipient ensures that the notification will be received by several people, which prevents dropped notifications when the lead Administrator is not available. The added advantage is that team members can be added to or removed from the distribution by changing the distribution list membership rather than reconfiguring the notification routine.

Alternately, a shared or "team" mailbox can be an excellent recipient for notifications; Administrators can share or schedule ownership of monitoring the mailbox for incoming notifications, and can file or delete notifications as action is taking, leaving only new or outstanding items in the Inbox.

Know the Environment

In any monitoring strategy, it is important that the Administrator be deeply familiar with the environment: the servers, their roles, the schedules of the routines, and so forth. Much of the "feedback" provided by the servers is irrelevant until it is put in its proper context. Is an alert for high CPU utilization a critical event that warrants immediate attention? Or is it merely a by-product of doing work? No matter what the monitoring approach is, some notifications will only require action if they occur at certain times of the day. If the CPU utilization increases dramatically at random times during the day, and Administrator may need to take action; the same notification at the start of the archiving window simply means that the server is busy doing its assigned work, and is not a need for concern.

Some monitoring strategies do allow the flexibility to run on a schedule, or alternately, suspend themselves at certain times. Review the options available in the chosen method and take advantage of flexible scheduling when possible, to cut down on "noise" notifications. Remember that each "noise"

notification de-sensitizes the Administrator to the notification process overall, making it less effective in the long run.

Also identify any key differences in server build or role, and adjust monitoring standards accordingly. If specific servers have smaller local drives than others, the standard Low Disk Space warning may not be suitable. Servers hosting archives required for legal e-discovery work may require a higher level of uptime, which would demand a more proactive approach than other servers. Identify servers with special requirements and plan the monitoring strategy accordingly.

Tracking Expected Activities

It may be helpful to document the expected timeframes of key routines, so that an Administrator can put notifications into context. Identifying the projected start and end times of scheduled archiving runs, provisioning runs, backups, and such can provide the context an Administrator needs to determine quickly if action is required. This chart can be used as a template for building an Activity Map.

	Midnight-6am	6-8 AM	8am-noon	Noon-4PM	4-6 PM	6-8 PM	8-10 PM	10PM-Midnight	Comments
Server1 Archiving									Expecting 12,000-20,000 items archived
Server2 Archiving									Expecting 12,000-20,000 items archived
Backups									
Provisioning									
Server1 Policy Sync									
Server2 Policy Sync									
Server3 Policy Sync									
Storage Expiry									
Move Archive									
Discovery Scheduled Searches									15-30 Nightly searches

Including the expected outcome of certain routines makes it easier to spot potential problems.

It may be useful to track activities on other, related systems such as Exchange Online Maintenance or backups, SQL maintenance routines, and so forth. Performance on the related application can decrease during these routines which can have a “knock on” effect on Enterprise Vault.

About Enterprise Vault Operations Manager

Enterprise Vault Operations Manager (EVOM) is a built-in feature of the Enterprise Vault application that allows for basic remote monitoring of the Enterprise Vault environment. When enabled, a background process, `MonitoringMiddleTier.exe`, runs on all Enterprise Vault servers and collects basic details such as task/service status (running, stopped, disabled, etc), journal mailbox queues, and basic performance metrics. These details are collected into the `EnterpriseVaultMonitoring` database and viewed by Administrators from the EVOM webpage, <http://MyEVServer/Monitoringwebapp>. Note that the web interface must be installed but that the background processes can continue to run without the web interface.

The topics in this whitepaper are meant to serve as a compliment to Enterprise Vault Operations Manager, providing additional flexibility and detail based on each customer’s unique environment.

More information about Enterprise Vault Operations Manager can be found in the version-specific `Installing and Configuring Enterprise Vault` guide, or at this technote and its related technotes: <http://www.symantec.com/docs/HOWTO36998>

Approaches

Reactive

In a reactive scenario, most monitoring is event-based; based on what happens, a notification or action occurs. This notification can be an event log message, Windows popup, text message, email or other alert. The notification can also include a post-event activity, such as an automated routine to apply corrective action.

For example, a company may have a pattern of losing Enterprise Vault functionality when a particular MAPI error occurs; the consistent solution may be to run the Microsoft FixMAPI utility and reboot the server. A reactive routine to perform those actions anytime the specific error is encountered could restore service before an Administrator has even had an opportunity to respond.

Reactive monitoring is among the easiest to implement because it is fairly easy to check for the existence of an event or condition. The downside, however, is that by the time the monitoring method is engaged, the condition has already occurred; the Administrator has no opportunity to “get out in front of” a potentially critical situation.

Some considerations in setting a reactive routine:

- *What is the problem condition and how is it identified?* When the problem manifests, what are its symptoms? Be as precise and granular as possible in defining the event; for example, if an Event Log event ID could have many different meanings, monitor for the event ID as well as keywords in the event body.
- *What is the risk of not taking immediate action?* If the risk is low or the problem affects only a few users, then an email alert or event log entry may be most appropriate; scripts that apply corrective actions are a better fit for problems that affect large groups of users. Use email alerts for single failed user indexes, and scripted transaction log truncates for SQL server disk space issues.
- *What is the normal corrective action taken? Can this be automated?* Most tasks can be automated through batchfiles, Powershell scripts, Visual Basic scripts, and so forth, then associated with the condition's occurrence.
- *Is the corrective action always identical and appropriate?* Some actions, such as restarting services or deleting temporary files, are valid in a wide range of situations and are good candidates for scripting; others, such as restarting stuck searches or purging MSMQ queues, could be easily scripted but may require an Administrator to determine if the action is appropriate.

Proactive

It is generally more difficult to develop a proactive monitoring approach; how can one check for a condition that does not yet exist? A proactive approach to monitoring can be based not on what has actually happened but on what is *expected to happen*. As an example, comparing historical archiving rates can help an Administrator determine what is expected to happen in a nightly archiving run; when actual numbers fall short, the Administrator may be able to expose a hidden performance problem.

To successfully create a proactive monitoring routine, the Administrator must first identify the normal, expected behavior in the environment. Some considerations:

- *What is expected to happen, and when?* Keep good records of schedules; include all key Enterprise Vault routines (archiving, provisioning, collections, Storage Expiry, and so forth). Also track related applications that may have an impact on Enterprise Vault such as backups, Exchange maintenance and SQL routines. Identify which services should be running during production and off-production times.
- *How will I know that it's happened?* If the scheduled routine does not write to the Event Logs, is there a log file that can be monitored, or a SQL entry?
- *What is a satisfactory outcome?* When monitoring performance and stability, it is not enough that services are running; the quantity of work and quantity of resources also matter. For example, identify a "normal" rate of SQL transaction log growth during the archiving window, and configure an email alert when the actual log growth exceeds the normal by 30%.

In most environments, a combination of reactive and proactive monitoring can yield a comprehensive approach.

Establishing a Baseline

To establish a baseline, document key metrics during periods of acceptable performance and stability. The baseline can then be used as a standard for comparison with future metrics. While each organization will have their own standards, some common criteria are:

- *Disk Queue Length:* Captured with Windows Performance Monitor, this metric captures how many requests are waiting for a disk read or write resource.
- *Available Memory:* Use Windows Performance Monitor to capture how much memory is being consumed by the Enterprise Vault server
- *Processor Busy Time:* This determines the amount of time the processor is kept busy and can be measured through Windows Performance Monitor

- *Disk Free Space*: Because Enterprise Vault services will shut down if a disk becomes too full, it is essential to monitor and track disk free space. Performance Monitor and WMI calls are two of the many ways to track free disk space.
- *MSMQ Queue Depth*: The number of items in the various MSMQ's can help document overall usage as well as user use patterns; this can be captured a number of ways including Windows Performance Monitor
- *SQL Transaction Log Size*: Use SQL syntax to capture the size of the transaction logs, which can grow rapidly based on activity
- *Archiving Rates*: For many customers, the daily/nightly archiving rate becomes the “gold standard” when evaluating performance. This is most easily tracked by SQL queries or through the Enterprise Vault Reporting component.

Once the baseline criteria is defined, collect data for at least one week to allow for the normal workload “peaks and valleys.” Set reasonable sampling periods; Performance Monitor can be configured to sample every 5-10 minutes, WMI calls can be executed every 5-10 minutes, and SQL queries can be broken out to daily or hourly numbers. Use these statistics to calculate an acceptable range, factoring in the expected activities; 85% CPU utilization may be an acceptable number *in your environment* during the archiving window but may be considered unacceptable during business hours.

Data Repositories and Tools

A wealth of information can be found in any Enterprise Vault environment. Understanding where certain data resides can help shape the monitoring strategy.

Windows Event Logs

The Windows event logs are one of the most accessible sources of information about the Enterprise Vault environment. They contain details of unexpected events such as problematic messages and network problems, as well as expected events such as services/tasks starting and stopping. The Enterprise Vault Administrator should periodically review the event logs and become familiar with those events that are common in the environment. From the list of common events, the Administrator can then determine which are *important* events.

Many third-party software products can create email notification based on specific events, or consolidate multiple event logs into a single interface but an Administrator may still need to configure a custom routine to fire an action (run a batch file,etc).

The most common challenge with Event Log monitoring is determining what events to monitor. Enterprise Vault writes a variety of warnings and errors, some of which carry greater meaning when put into context. Some events should always be considered critical, such as services stopping unexpectedly; others are simply describing common conditions such as warnings about users who are not found to be in a Provisioning Group. Most events fall somewhere in between, and can be more or less important only in some environments or at some times of the day.

Some basic questions can help determine the importance of an event:

- *Does the event indicate a service outage is likely or has already happened?* Events regarding services that stop unexpectedly, lost database connectivity, or lost access to storage frequently indicate or lead to lost functionality. These events generally warrant quick attention.
- *Does the event indicate a problem with a single resource?* These events tend to have moderate impact: lost functionality for some, but not all users. Examples are tasks that fail due to lost connectivity with the Exchange server, and individual indexes that have failed.
- *Does the event require immediate attention?* Warnings regarding user mailbox quotas or permissions generally don't disrupt functionality for a broad user community and can be excluded from monitoring or given a lower priority. Alerts, notifications and actions should be reserved only for those events that need attention right away.

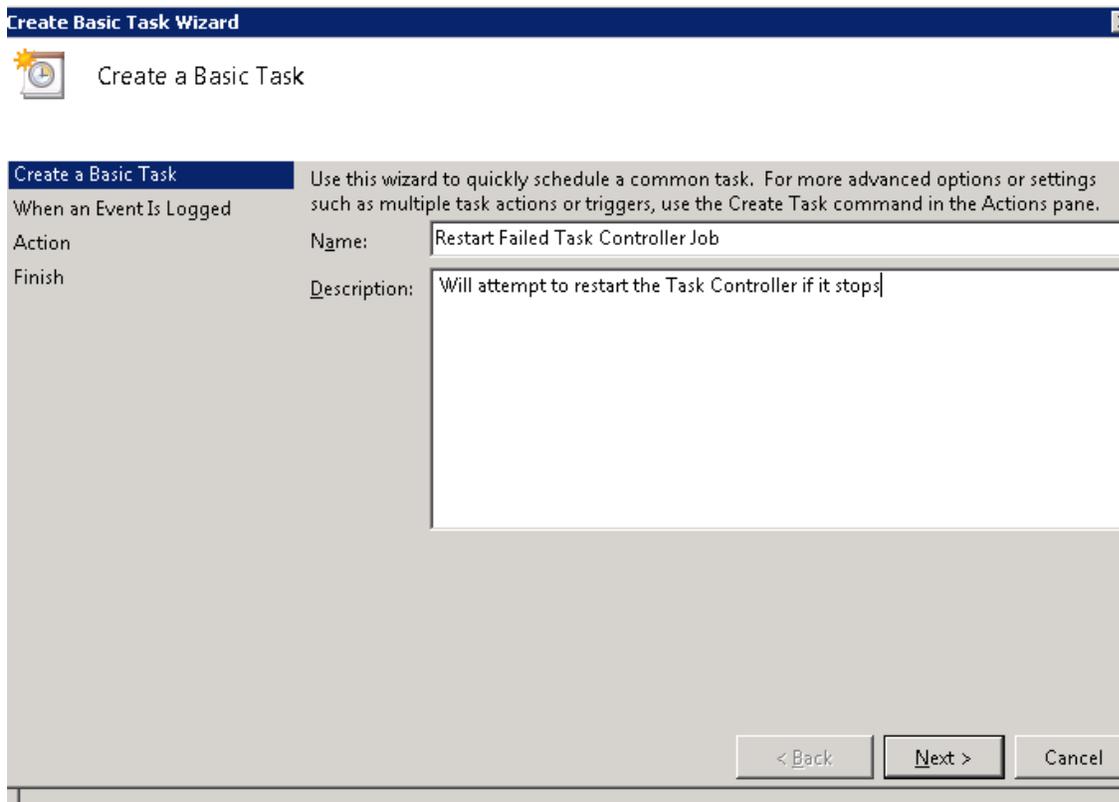
It should also be noted that a monitoring strategy will not eliminate the need for periodic review of event logs. Low-impact warnings must still be investigated, and the overall Event Log messages should be reviewed for patterns that may not be represented by any single event.

Setting Up Notifications from Event Logs

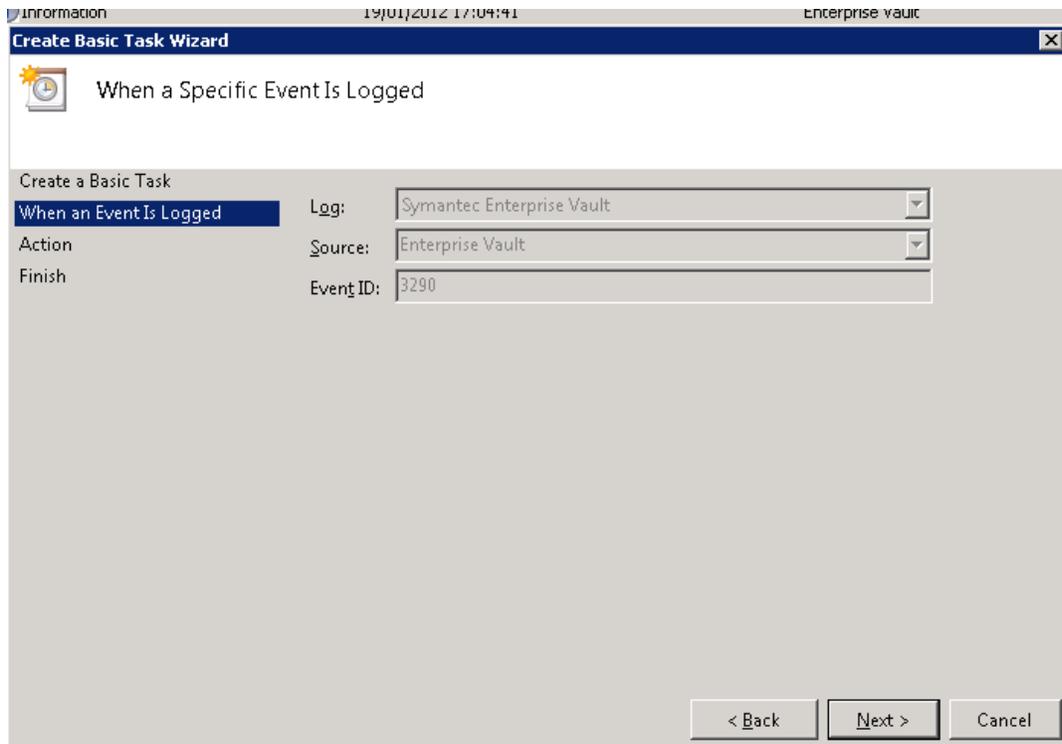
The steps will vary with the Windows operating system version and the preferred scripting language; a sample script will be provided in the Appendix of this document. Windows 2008 Server operating system introduces a GUI-based interface where an Administrator can configure a Task to be associated with a specific event. The Task can be to send an email, run a program, or display a message on the computer's desktop. The configured Task will be associated with Windows Task Scheduler. As an example, the Administrator may wish to create a routine that starts a particular service if an event indicates that the service has stopped.

This example assumes that a batchfile to start the service already exists. A sample batch file can be found in the Appendix of this document.

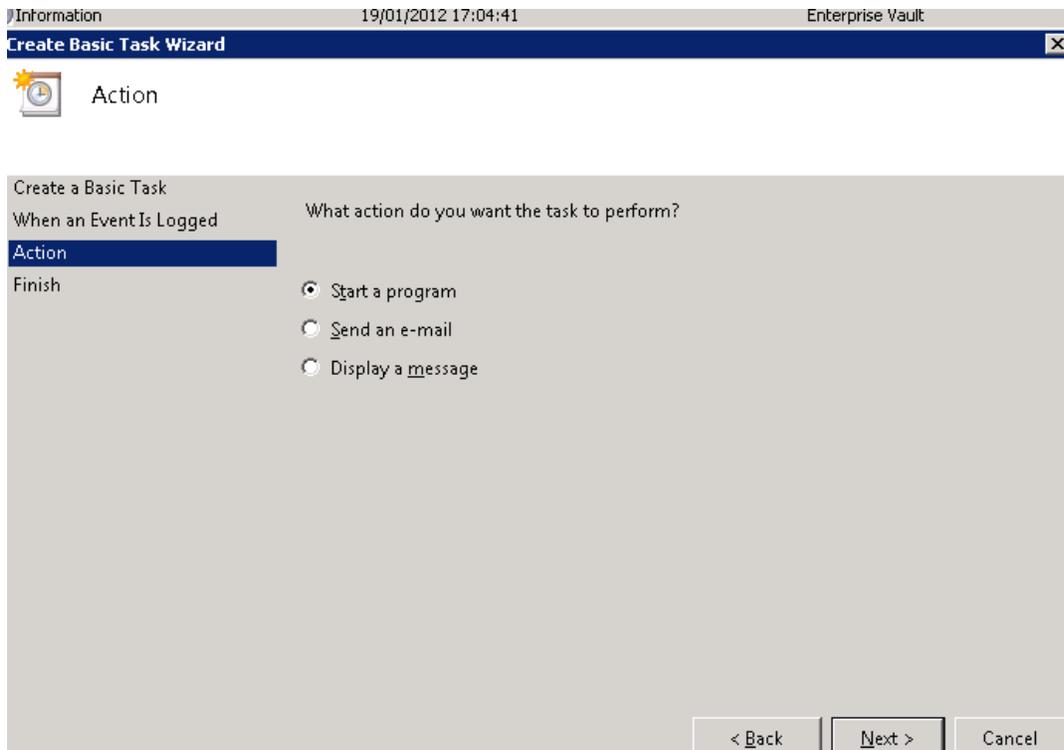
1. Create, test and store the batchfile
2. Access the server Event Viewer and locate the relevant event



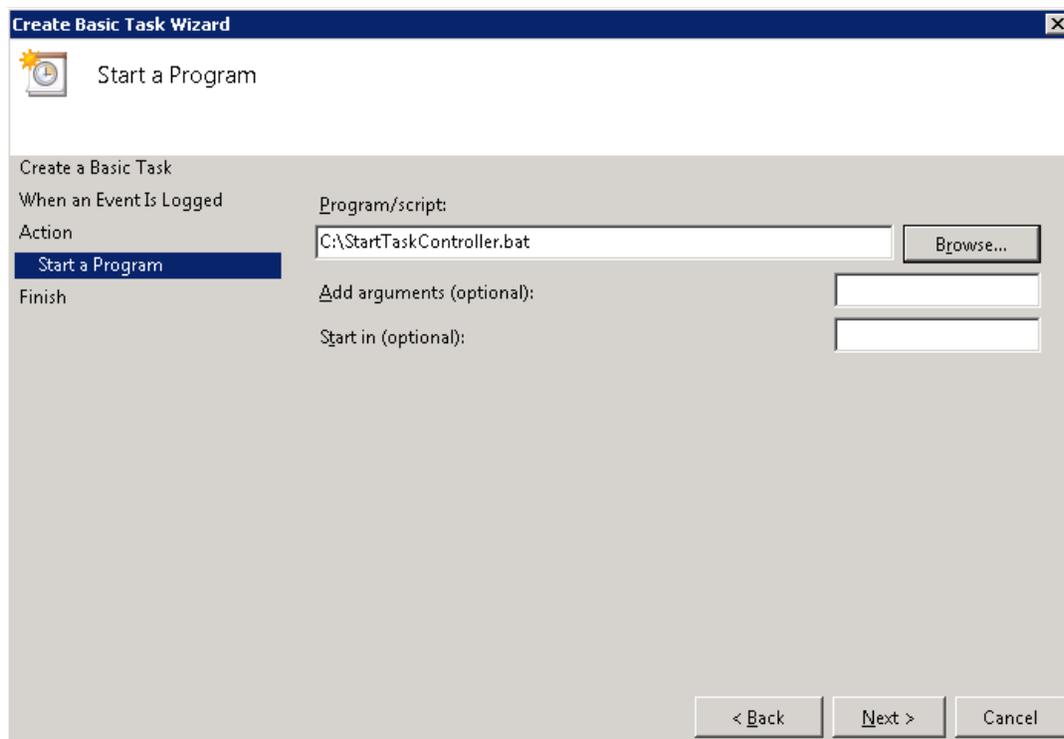
3. Right-click the event and select Attach Task to This Event. The Create Basic Task Wizard appears.
4. Populate the name and a description of your routine, then click Next



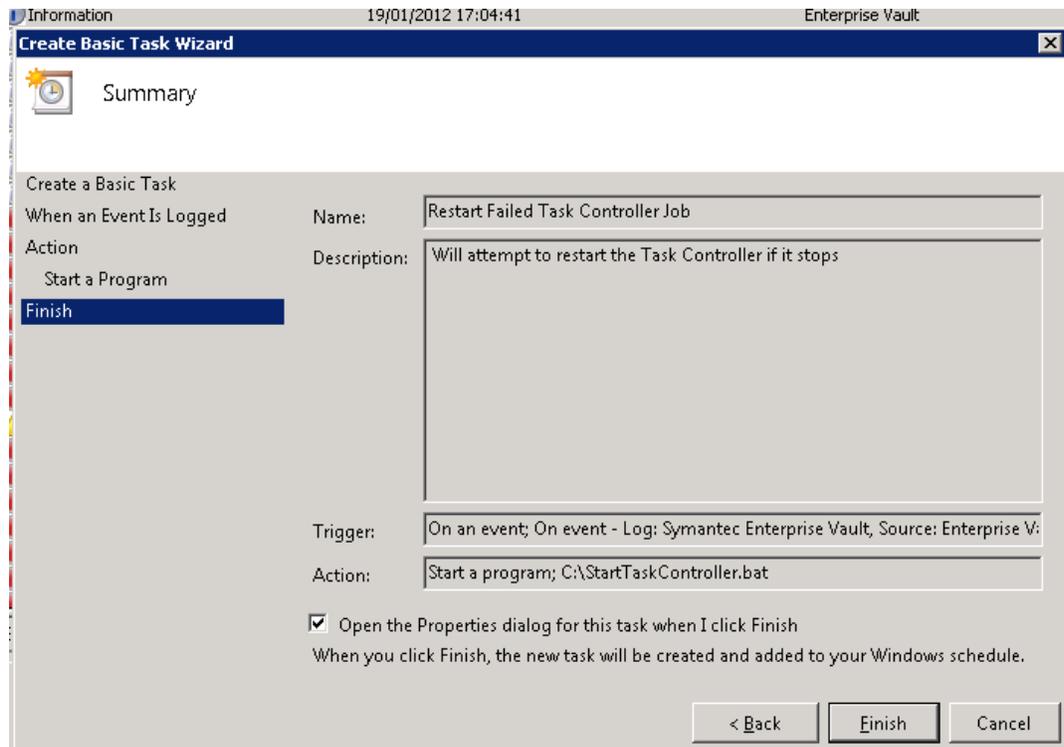
5. Notice that the “When an Event Is Logged” dialog is pre-populated based on the selected log and event. No changes can be made to this page of the Wizard. Click “Next” to continue.
6. Select the action you wish to take upon encountering the event. In this example, select Run a Program.



7. Browse for the desired program, in this example the StartTaskController.bat



8. Verify the settings as configured. An Administrator can also use the checkbox to open the Properties of the Task, to configure advanced options such as how many times the Scheduled Task should attempt to run, should it not be successful on the first attempt.
9. Click Finish to exit the Wizard.

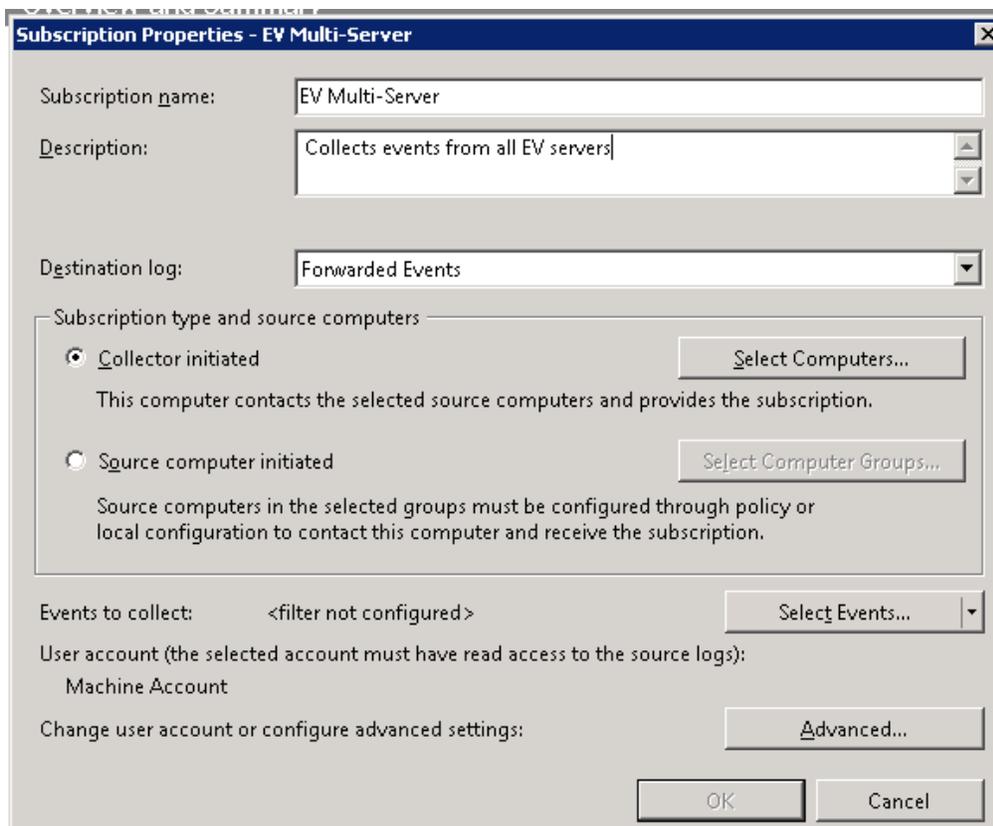


Note: Once configured, the Task(s) can be exported from the server and imported to additional servers.

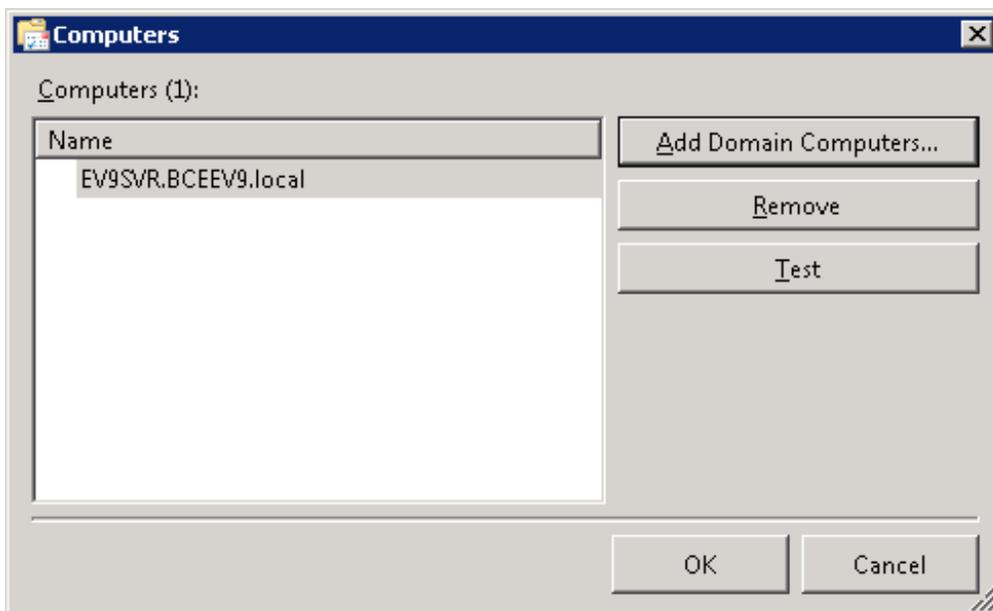
Managing Multiple Event Logs

Many Administrators routinely review Windows Event Logs to find warnings and errors. This is often a manual process that can become tedious and time-consuming, especially in larger environments. There are many third-party softwares that combine Event Logs to a central repository; Powershell, Visual Basic and other scripts can be used to do the same. It is important that an Administrator implement a method for collecting the Event Logs. Windows Server 2008 introduces Event Log Subscriptions, with which an Administrator can configure multiple remote event logs to write to a central repository.

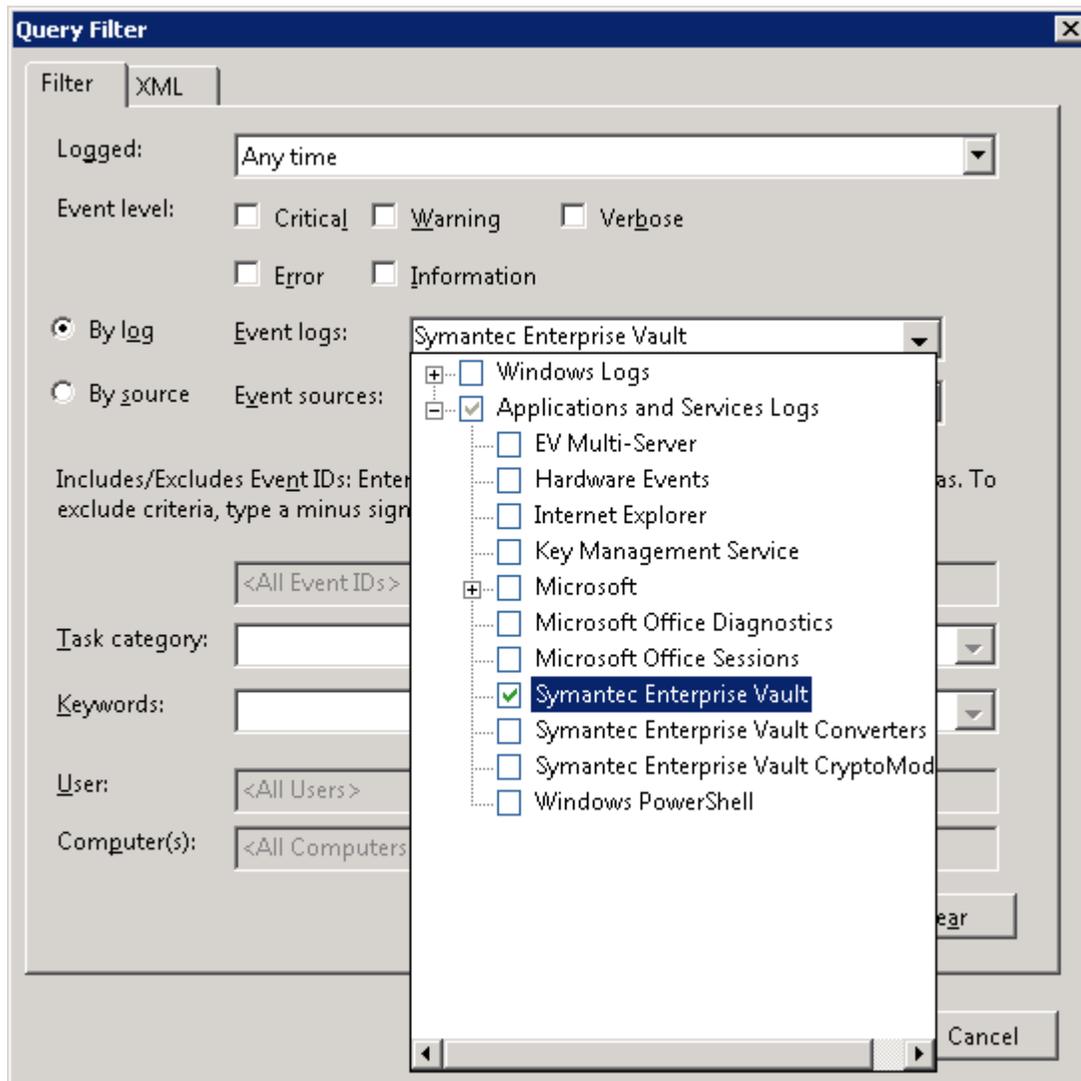
To create an Event Log Subscription, launch the Windows Event Viewer, right-click Subscriptions→New Subscription. The Create Subscription dialog appears. Fill in the name of the subscription and all desired settings. Note: it is possible to create a custom Event Log repository to hold these events if you do not wish to use the generic Forwarded Events log; consult the Microsoft documentation for version-specific instructions.



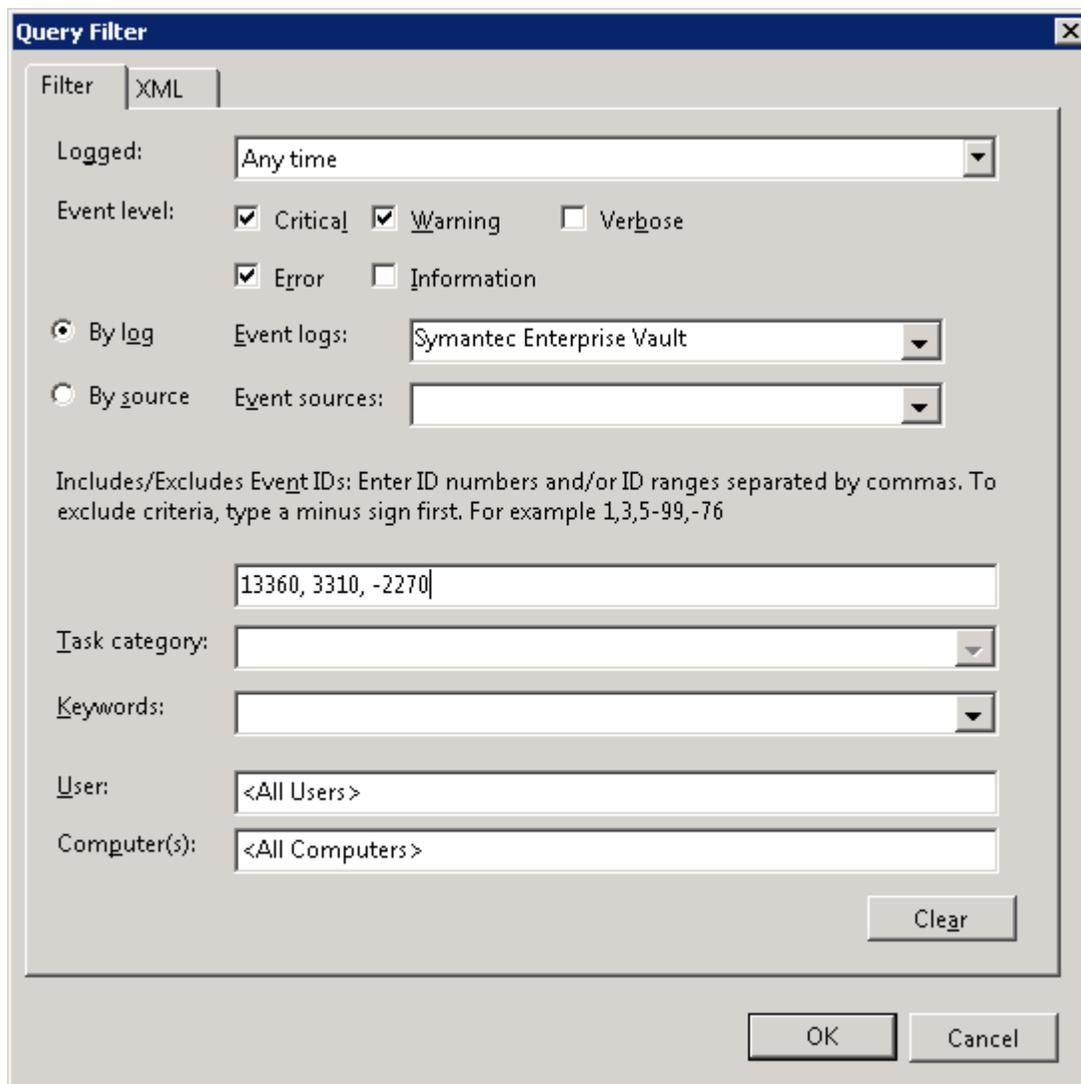
Selecting the subscription type “Collector Initiated” will allow the Administrator to define the servers to be queried; the server hosting the subscription will poll the remote computers periodically for events to collect.



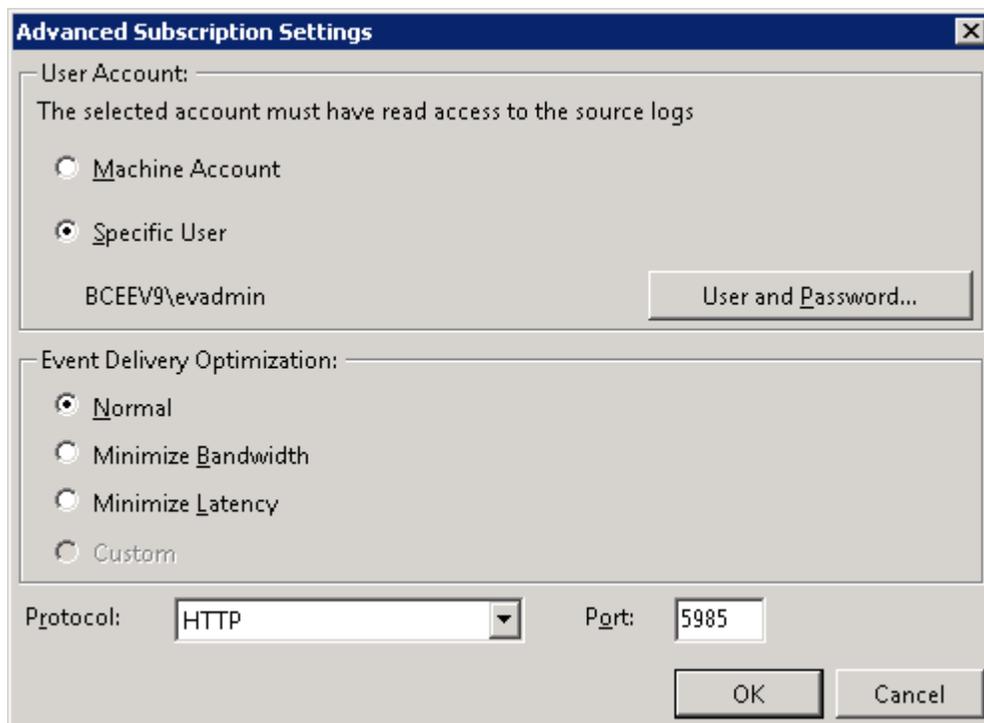
Use the Events to Collect section of the Add Subscription dialog to define which events will be forwarded from the remote server. You can filter by entire log (i.e. collect the Enterprise Vault log but not the System log), Event Category, Event Source, information/warning/error, and Event ID.



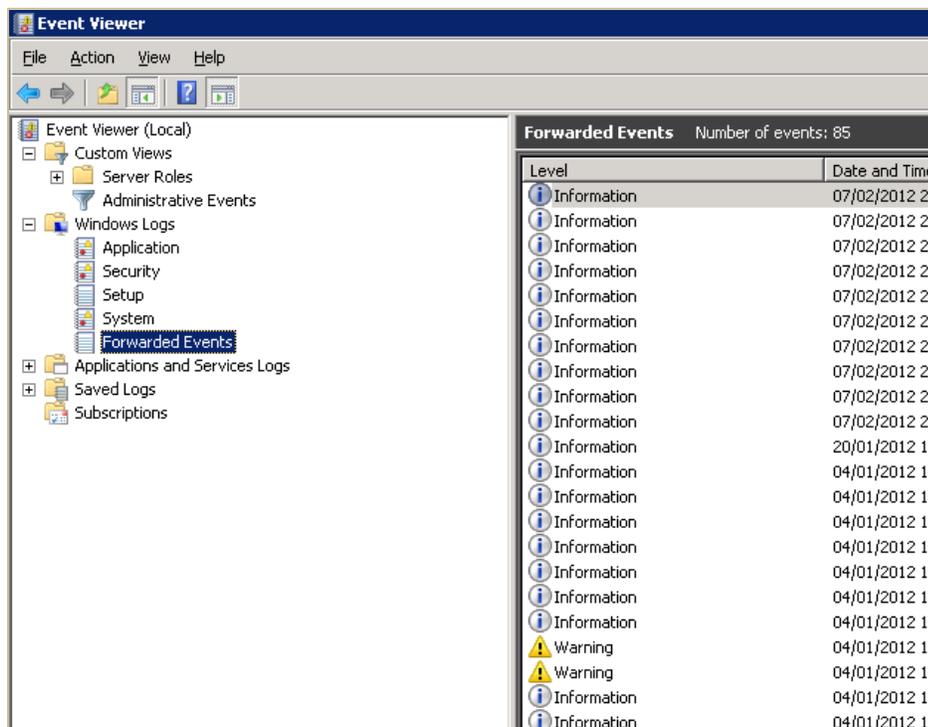
Include specific Event ID's by entering them on the line provided; discard event ID's by placing them on the same line, starting with a – (minus sign). For example, an Administrator may wish to discard warnings about users' Exchange quotas.



Use the Advanced Options to configure the subscription to run under a particular network account such as the EV Admin account.



When the Subscription is active, events from the targeted servers will appear in the Windows Logs→Forwarded Events (or whichever log was selected as the destination).



Note: some Windows services and configurations may be required before Event Log Subscriptions can be configured; please review Microsoft product documentation for details.

Windows Performance Monitor (PerfMon)

Windows Performance Monitor is an operating system utility allowing the Administrator to view and record a wide array of performance-related metrics. Some example metrics are disk space thresholds, disk queue lengths (which can indicate a bottleneck when reading from or writing to the disk), or free memory space. An Administrator must determine which metrics are of importance before beginning to collect Performance Monitor data.

Once configured, Performance Monitor settings can be exported as a HTML-based template, to be imported to other servers. This can save time when monitoring multiple Enterprise Vault servers, and can be an effective way to recreate the desired settings when hardware is replaced (either planned or as part of a disaster recovery). An Administrator can also configure multiple Performance Monitor templates, each with different settings, to be used for troubleshooting purposes. A Performance Monitor template configured to collect deep detail about memory utilization can be imported to quickly capture information about a suspected memory leak.

Performance metrics are highly environmental. Using the Disk Read/Write Queue Length as an example, Microsoft recommends in many documents that the ideal queue depth is less than two *per spindle*, but the spindle count varies with the storage hardware and configuration. Defining the actual acceptable performance metrics will be outside the scope of this document; an Administrator should review guidelines for the installed hardware and operating system, then collect a baseline to determine what is appropriate for the environment. Commonly used Performance Monitor counters include Disk Read/Write/Combined Queue Length (excessive queuing indicates a potentially undersized processor), Processor % Processor Time (indicating how much of the time the processor spends actually executing work), and Memory Available Mbytes (indicating how much of the server memory is actually in use).

There are two primary methods of utilizing the Performance Monitor: by configuring alerts, and by configuring historical logging.

Working With Performance Monitor Statistics

Historical logging is useful when the Administrator wants to evaluate overall performance, especially in relationship to the design and architecture. By capturing key Performance Monitor metrics, the Administrator can review a long-term picture of how the Enterprise Vault server responds to the requested workload. Comparing historical information periodically can help the Administrator identify a performance baseline, and alert an Administrator to upward trends in usage (which could, over time, translate to slower performance if the hardware is not sufficient for the future load).

Historical Performance Monitor data can be captured to various formats; the native format is .blg and is easily analyzed in the Performance Monitor tool or by using Microsoft's Performance Analysis and Logging (PAL) tool. Many Administrators prefer to collect data in comma-separated value (CSV) format, to open and analyze in Microsoft Excel. This technique can be very useful when looking for trends; for example, by opening the CSV file in Microsoft Excel and converting to a chart with date and time references, the Administrator can determine if the statistics follow an expected pattern. High processor utilization during the archiving window but not outside of it, for example, is not indicative of a problem; one would expect an Enterprise Vault server to be under heavy load when it is actively processing hundreds or thousands of items.

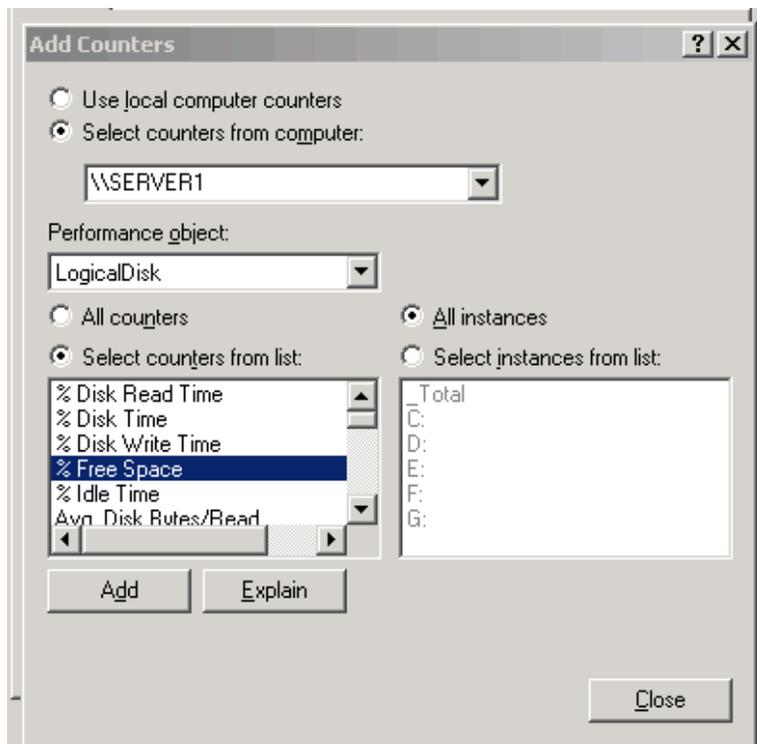
Configuring Performance Monitor Alerts

By configuring alerts, the Administrator can be informed of critical conditions, such as insufficient disk space and other issues that require immediate attention. In many cases, the alert can be configured through the Windows Performance Monitor GUI; however, the steps will vary with the operating system version. The Administrator can choose the alert action: write to the Windows Event Logs, run a program, or send a message via the Windows Messenger Service. The Administrator can also start an additional set of Performance Monitor counters when a specific condition is met; for example, if available memory falls below a certain threshold, start an additional set of counters which collect deeper detail about memory utilization.

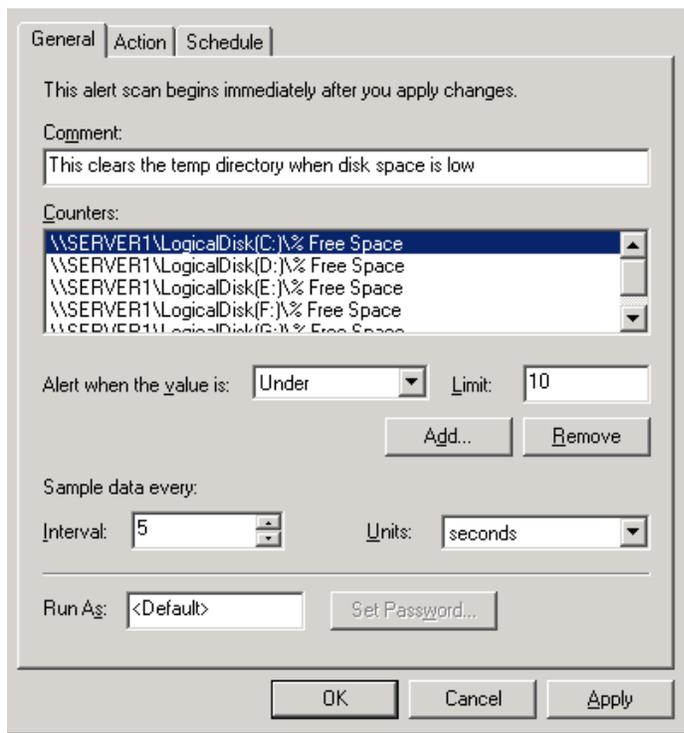
In this example, the Administrator wishes to clear out temporary (temp) files if the free disk space falls below a certain threshold. This example, using Windows 2003 Server, assumes that the script to clear the temp directory is already written; a sample script can be found in the Appendix of this document.

Note: The Enterprise Vault Admin Service will also monitor free disk space, and will shut services down if disk space is too low. An outage may be prevented by automating the "housekeeping" work.

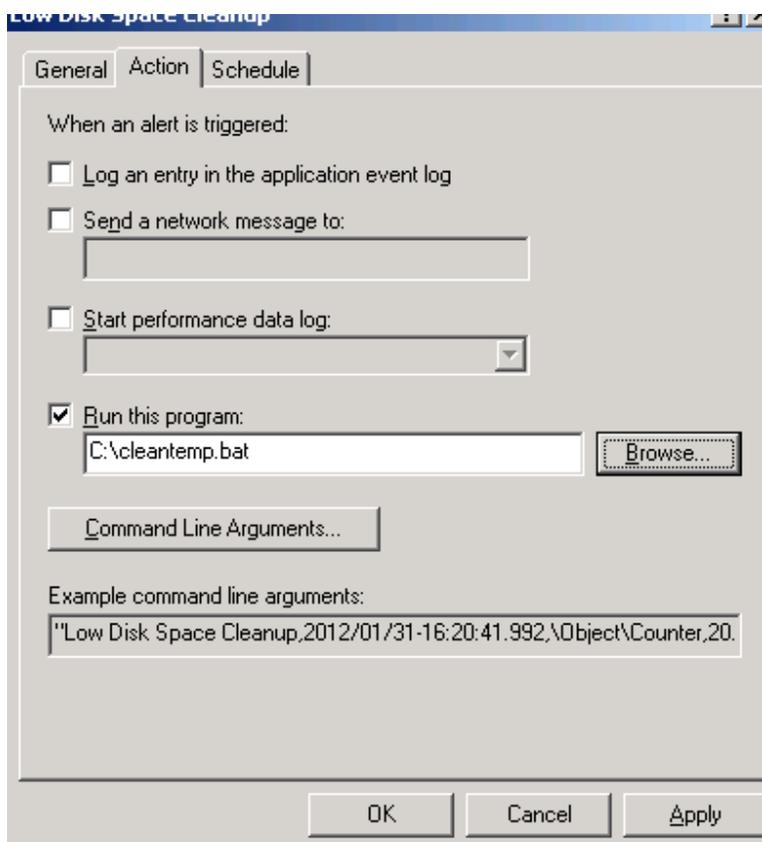
1. Create and test the batchfile
2. Open Windows Performance Monitor
3. Expand Performance Logs and Alerts→Alerts; right-click Alerts, select New Alert, and name the Alert
4. The Configure Alert window appears. Provide a description of the alert, then click the Add button to begin adding counters



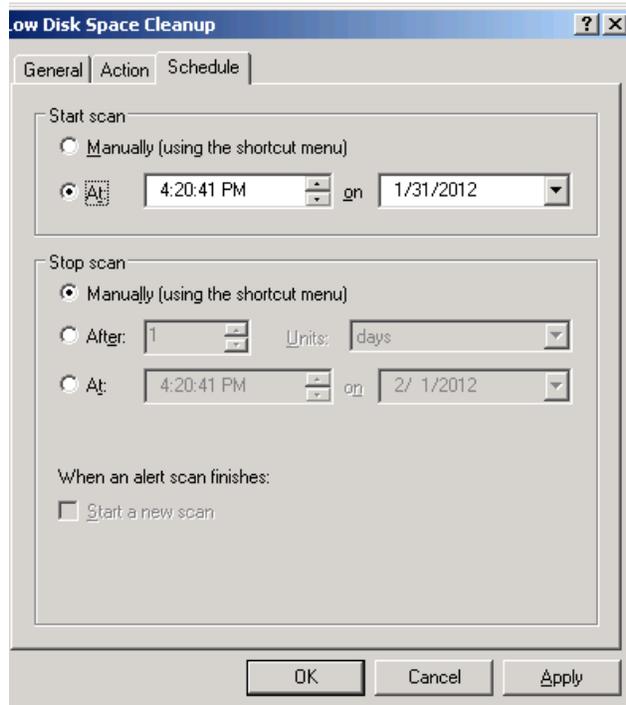
5. Select the Logical Disk object, then the % Free Space counter. By selecting the “All Instances” radio button, the alert can be configured on all logical disks; or use the “Select Instances from list” radio button and configure the individual disks. Click Add to commit the counters then Close to return to the main Alert configuration dialog.
6. Highlight each instance and configure the alert threshold in percentage of free space, selecting Under from the drop-down list.



7. Configure an appropriate scan interval and the user credentials under which to run the scan
8. Click the Action tab and select "Run a Program" then browse to the batch file created earlier. Supply any relevant command-line switches



9. Click the Schedule tab to supply an optional schedule for running the scan. Alternately, start the Alert routine immediately and let run indefinitely from the main Performance and Alerts window.



Once configured, the settings can be exported to a HTML file by navigating to Windows Performance Monitor. Expand Performance Logs and Alerts→Alerts, locate the desired configuration in the right-side pane, right-click the Alert and select Save Settings As. Save the file in HTML format.

The HTML file can later be imported back to the same server (for quick re-configuration in a hardware migration or disaster recovery) or can be edited to reflect different server names, allowing the Administrator to quickly apply an identical configuration to multiple servers. Note that any batchfile to be run must exist in the same path on each target server, and additional drives must be added (either by editing the HTML file before import, or manually through the GUI) and excess drives must be removed for the Alert to function properly.

To import the HTML configuration, in the Performance Logs and Alerts→ Alerts interface, right-click Alerts then select New Alert Settings From and browse to your HTML template file.

Scheduling Performance Monitor Using Logman

Some performance metrics are relevant only in the context of the expected workload; this is especially true in the case of Performance Alerts which trigger actions based on thresholds. An alert regarding CPU passing 80% utilization would be relevant in many environments during the workday, but would be irrelevant in the same environment at night when the archiving window opens. When the additional workload of archiving is added to a server, it is expected and normal to see the CPU utilization jump. Likewise, an alert indicating over 5000 items in the MSMQ A5 queue would not be considered an indicator of a problem if it is triggered at the start of the archiving window.

A successful performance alerting strategy would factor in expected peaks in load based on what is happening at a given time of day. This helps keep Administrators from becoming de-sensitized to the alerts, and cuts down on time spent responding to false alarms.

An Administrator can use the Windows Logman utility to manage starting and stopping configured Perfmon collectors. The syntax can be built into a batchfile which can then be run on a schedule. The sample syntax below assumes that the Perfmon collector set has already been configured and tested.

To list the names of configured collector sets, open a command prompt and issue the following command. The list of configured collector sets will be displayed.

Logman query

```
C:\>logman query
Collection                                Type                                Status
-----
msmq                                       Counter                             Stopped
MSMQ Alert                               Alert                               Stopped
Test Automation Alert                   Alert                               Stopped
System Overview                          Counter                             Stopped
Low Disk Space Cleanup                   Alert                               Running

The command completed successfully.
C:\>logman start "msmq alert"
The command completed successfully.
C:\>
```

The name of the collector set can then be included with the logman command and an instruction to start, with the following syntax. If there are spaces in the collector set name, enclose the collector set in quotes.

Logman start "MSMQ Alert"

```
The command completed successfully.
C:\>logman start "MSMQ Alert"
The command completed successfully.
C:\>
```

These simple commands can be written to a batch file which would then be called with Windows Task Scheduler. An Administrator can define two different collector sets, one relevant to work/production hours and another tailored for the more intense archiving window, and schedule them accordingly.

More recent versions of the Windows operating system include additional "verbs" for use with Logman including export/import (to easily copy the collector sets between servers) and remote functionality. Logman can also load desired counters from a config (.cfg) file, allowing an Administrator to prepare performance monitoring templates to be used when troubleshooting. Review the Microsoft Windows Server documentation for your version of Windows for the most up-to-date functionality.

SQL Databases

Considered the “main brain” of Enterprise Vault, the various SQL databases contain detailed information about Enterprise Vault servers, users, archived items and more. By mining these values and statistics, the Administrator can begin to develop realistic expectations for how Enterprise Vault *should* work in their environment. Once a baseline is established, the Administrator can then compare actual activity, and focus on any deviations.

Alternately, using SQL queries, an Administrator can gather specific details of a user archive and examine for known problematic conditions. A Discovery Accelerator Administrator can likewise review the status and statistics of all searches to identify problematic ones.

The EnterpriseVaultMonitoring database is an excellent data repository as the Enterprise Vault Monitoring processes can gather relevant metrics into the database, making it easy for an Administrator to query with simple SQL scripts. Metrics such as the status of services and tasks can be queried directly from the database without writing complex WMI or MAPI calls.

Mining SQL Data

There are a variety of approaches to accessing the data stored in the SQL database. An Administrator can simply run any SQL query for an on-demand, point-in-time look at a specific metric; the same SQL queries can be configured as part of SQL scheduled jobs. For example, a known performance problem can occur when a specific attribute is missing or malformed in an archive’s record in the database; a query can be run daily and an email sent to an Administrator if any problematic archives are discovered.

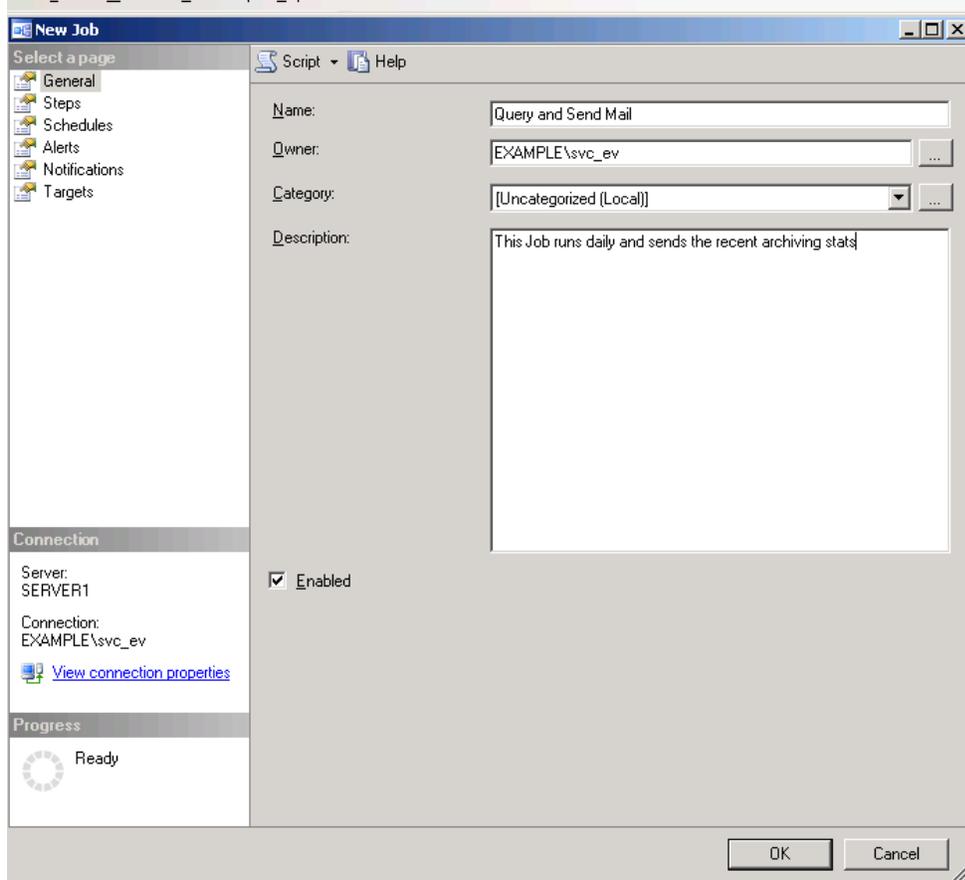
An Administrator can build custom reports using Microsoft SQL Reporting Services (SRS). The developed report can collect user and archive statistics, to be used by helpdesk staff in routine user support calls.

Using SQL Jobs

In this example, an Administrator wishes to proactively check for signs of performance or connectivity problems by monitoring nightly archiving rates. The Administrator has already determined what a “normal” night’s work would total, and wants to receive a daily email reflecting the archiving activity.

This example assumes that a SQL Mail Profile already exists on the SQL server, and that the SQL Agent Service is started. This example is demonstrated with SQL 2005; consult the Microsoft documentation for version-specific steps for your SQL server. This also assumes that the relevant query has already been written and tested.

1. Open SQL Management Studio, connect to the desired SQL server, and expand SQL Agent→Jobs
2. Right-click Jobs and select New Job. The New Job dialog appears.



3. Configure the name, owner, category (if any) and a short description of the job. Use the Enabled checkbox to control if the job is to be engaged immediately.
4. Click into the Steps tab and configure each step of the routine.
5. To begin the routine by running a query, name the Step, then select Transact-SQL from the drop-down box and enter the query in the window provided. Use the Parse button to test the syntax. A sample query is provided in the Appendix.

The screenshot shows a configuration window for a script step. The window title is "Script" with a "Help" button. The "Step name" is "Query and Send Mail". The "Type" is "Transact-SQL script (T-SQL)". The "Run as" field is empty. The "Database" is set to "msdb". The "Command" field contains the following Transact-SQL script:

```
USE MSDB
EXEC sp_send_dbmail
  @profile_name = 'EVMailer',
  @recipients = 'tamara.kramer@example.com',
  @subject = 'Nightly Archiving Rate',
  @body = 'Last night numbers are as follows',
  @execute_query_database = 'EVEexchangeVaultStore',
  @query = 'USE EVEexchangeVaultStore Select CAST(SUM(Itemsize/1024) AS Bigint) as Total_MB,
           Count_Big (idtransaction) as ItemCount
           From Saveset Where ArchivedDate < (GETDATE() - 1)'
```

On the left side of the Command field, there are five buttons: "Open...", "Select All", "Copy", "Paste", and "Parse".

Note: the Transact-SQL that sends mail MUST be run from the context of the MSDB. This can be selected from the drop-down database list and can also be defined at the start of the routine. Running in the context of any other database will cause the procedure to fail. To run the mailer routine but collect statistics from any other database, include the USE clause in the @query line of the routine, and define the desired database(s) on that line.

1. Configure as many additional steps as desired, taking care to place them in the correct order.
2. Click into the Schedules tab and configure the desired schedule.

New Job Schedule

Name:

Schedule type: Enabled

One-time occurrence

Date: Time:

Frequency

Occurs:

Recurs every: day(s)

Daily frequency

Occurs once at:

Occurs every: hour(s)

Starting at:

Ending at:

Duration

Start date: End date:

No end date

Summary

Description:

- All other details and value are optional. At this point you can save the job and test it by right-clicking the job → Start Routine.

IIS Logs

The IIS logs contain a wealth of information about the Enterprise Vault functions that rely on IIS components. User-facing search functions, vault cache synchronization, and user restores are a few of the transactions that call IIS components. By understanding the various IIS functions and resources, the Administrator can collect valuable metrics: who is accessing Enterprise Vault data, how is it being accessed (upload or download), how frequently, throughput rates, and patterns of failure or success.

Some of the more commonly accessed IIS resources are as follows:

- Download.asp, DownloadContent.asp and DownloadContent.aspx** : These pages are called to process user download requests, with Download.asp being used for single-item downloads from the various web applications such as OWA and search tools, and the two DownloadContent pages being used when building and delivering Vault Cache content to end users.
- Search.asp**: this ASP page accepts and processes user requests to search the archives online (as with the various search tools, not when searching the Vault Cache with Outlook native tools or Windows Desktop Search)

- **ArchiveExplorerUI.asp**: this ASP page builds the user-specific Archive Explorer view. Tracking its usage can demonstrate the users' reliance on (or lack of use of) the Archive Explorer.
- **DeleteO2k.asp**: this ASP page is the entry point for user delete requests.
- **Upload.asp**: this ASP page is the entry point for user upload requests, namely items that are archived manually *through the Virtual Vault client*.
- **Slot.aspx**: this page is called as part of the Vault Cache Sync routine; this is a call to request a slot to synchronize an archive.
- **FullSync.aspx** and **IncrSync.aspx**: when synchronizing metadata (Vault Cache) these resources will be called, with FullSync.aspx being the resource used to perform a full metadata sync, and IncrSync.aspx being used to perform an incremental sync from the user's Transaction History present in the SQL database.
- **GetSlotWithServer.aspx** and **GetIncrSlotWithServer.aspx**: these ASPX pages will be called to initiate a connection (slot) to perform Vault Cache content synchronization.

Other details of note in the IIS logs will be the bytes sent/received, time taken, and any IIS-based success or error codes. While many of these details will be extremely relevant in troubleshooting scenarios, from a monitoring perspective, the number of visits, number of unique users, and/or the time ranges of peak activity can be the most relevant metrics.

Understanding when and how frequently users are using the various IIS resources can tie into a successful monitoring strategy; for example, identifying that users rely heavily on the ASP resources between 10am and 3pm, based on attempts to access the various .asp pages, would lead an Administrator to configure a performance alert on the counter Active Server Pages-> Requests Queued. While the threshold would be an environment-specific number, longer queue depths indicate slow performance.

Mining the Enterprise IIS Logs

By themselves, the IIS logs can be very large and cumbersome to navigate. IIS logs are particularly well-suited for use with the Microsoft Log Parser utility. The Log Parser utility lets an Administrator pass queries against the IIS logs (as well as many other types of logs) using a syntax that is based on Microsoft SQ to mine relevant information with a minimum of effort.

The Log Parser utility is run from the command line. When installed, a Samples folder is also created, holding syntax samples for many commonly-used IIS queries. The syntax can support a wide range of queries.

```
C:\Program Files\Log Parser 2.2>logparser -i:iisw3c "select time, s-ip, cs-method, cs-uri-stem, cs-uri-query, sc-status, cs(User-Agent) from c:\windows\system32\logfiles\w3svc1\ex080227.log" -q on >c:\testlog.txt
C:\Program Files\Log Parser 2.2>
```

The query example above is as follows:

```
logparser -i:iisw3c "select time, s-ip, cs-method, cs-uri-stem, cs-uri-  
query, sc-status, cs(User-Agent) from  
c:\windows\system32\logfiles\w3svc1\ex080227" -q on >c:\testlog.txt
```

The syntax indicates that the input file will be an IIS/W3c log. The string within the bracket is structured similar to Microsoft SQL: “select the IP address, method and status from the log file ex080227 in the listed path.” The uri-stem is also included in the SELECT statement; the uri-stem is the beginning of the remote request which in most cases includes the resource name (i.e. what ASP page the user is visiting). This example also includes an instruction to write the output to a text file, as indicated by the > character; this can also be specified through the Log Parser native syntax.

The syntax can be further refined by including *filters*: other conditions that the logged request must meet before being added to the Log parser output.

To filter by a specific page (for example, the download.asp page), include the following string: where cs-uri-stem like ‘%download.asp%’

To filter by a specific user, include the following string: cs-username like ‘%Joe.User%’

The % sign, as in Microsoft SQL, acts as a wildcard. The placement of the % is critical as it indicates the point at which an exact string match must begin or end. Placing the desired text string in between both a leading and a trailing percent sign means that the desired string can be found anywhere on a given line, with any text preceding or following it.

Microsoft Message Queuing (MSMQ)

The data stored in the MSMQ should be constantly changing: work requests enter the queues, are processed along the different stages/queues, complete and are removed from the queues. By examining what is in the queues at a given time, the Administrator can get a glimpse of the health of the Enterprise Vault system.

Prior to monitoring the Enterprise Vault-related MSMQ’s, the Administrator must fully understand the type of work executed from each queue and the times during which that work is scheduled to take place. As an example, items processed during the scheduled archiving run executes from the A5 queue; therefore, work requests that are not completed prior to the end of the archiving schedule should be expected to remain until the next archiving window. In other words, simply having work in the A5 queue for 8 hours does not always indicate a problem; but a problem may exist if the same number of requests were found in the A2 (manual archiving request processing) queue.

An Administrator can learn more about the role and function of each MSMQ queue by reviewing the version specific Enterprise Vault Administrator’s Guide.

Using the Typeperf Command to View MSMQ Statistics

Identifying a buildup of items in the MSMQ can give the Administrator an early warning of a developing problem. The Windows Typeperf command can give the Administrator up-to-the-minute statistics by writing data collected from the Windows Performance Monitor subsystem to the command window or to a log file.

To effectively use the Typeperf command to view current MSMQ depth, some pre-work is required. The Administrator must gather the relevant object and counter names from each server to be monitored, and build a config file which will ultimately control what Typeperf collects. This syntax can be used to dump all the counters and instances associated with the MSMQ Queue object:

```
Typeperf -qx "MSMQ Queue" -s MyEVServer -o c:\MyEVServer.txt
```

In this example, replace MyEVServer with the name of the EV server whose counters you wish to dump. The `-qx` switch indicates "dump the instances available for the MSMQ Queue object" while the `-s` indicates the server (if left blank it will dump the local host) and the `-o` indicates the output file. Note that the output file will be overwritten on subsequent runs; use a unique filename for each server.

Once the counters and instances for each server have been dumped, use Notepad or a similar text editor to combine all server information into a single text file. This makes up the config file that can be used to run Typeperf against multiple servers at the same time. The syntax will be as follows:

```
Typeperf -cf c:\MyServers.txt -sc 1
```

In the above example, the `-cf` switch indicates which config file to use, while the `-sc` switch indicates to take one sample only.

```
Typeperf -cf c:\MyServers.txt -sc 1 -o c:\AllMSMQ.csv -y
```

In the above example, the `-o` is an instruction to write the results to an output file and the `-y` is an instruction to answer Y(es) to any prompts, as there will be a prompt to overwrite the output file if one already exists. This syntax is useful if the Administrator wants the output written to a file that can be reviewed or emailed via script.

Additional details about the Typeperf command can be found at <http://technet.microsoft.com/en-us/library/bb490960.aspx>

Client Registry

Customers running the Vault Cache component can find information about the cache status by monitoring the user registry; all customers running the Enterprise Vault client (with or without Vault Cache) can gather details about the client itself, such as the build of the client. The most effective way to gather this information is by using SMS or another software deployment suite to capture the relevant registry keys; however, machine-based and user-based login scripts are also options.

To obtain the installed client version, which is useful in planning software rollouts, collect the following registry keys:

```
HKEY_LOCAL_MACHINE\Software\KVS\Enterprise Vault\Install
```

The DesktopClientVersion key records the major and minor version of the installed client kit, while the InstallPath key records the desktop install path.

To determine the last good sync of the Vault Cache's Content Cache, collect the following registry key:

```
HKEY_CURRENT_USER\Software\KVS\Enterprise Vault\Client\User-specific  
string\OVCCLastCompleted
```

Note that the *User-specific string* will be unique to each user and reflects the Exchange attribute PR_RECORD_KEY. This attribute is not stored in the Enterprise Vault databases. The Administrator can use LDAP queries to resolve this attribute for each user, or can fetch the parent key for each user.

Automating Registry Export and Collection

Software deployment is automated in many environments. If there already is an automated software management solution in place, consult the product vendor for options and methods to retrieve registry keys from client machines.

In some situations, a simple login script can be put in place which would write the relevant registry keys on behalf of the user. Registry keys can be written to the local machine, or to a central repository for administrative review.

The basic syntax to export the registry from the command line is as follows:

```
REG EXPORT "HKEY_CURRENT_USER\Software\KVS\Enterprise Vault\Client"  
c:\OutputLogs\EVClientReg.txt
```

The REG EXPORT command should be immediately followed by the key to export, enclosed in quotes, then desired output file name and path. To export to a sub-directory, the sub-directory must already exist. The command will export the selected key and all sub-keys. Because the next-level key contains the user-specific PR_RECORD_KEY variable, exporting the Client key with its subkeys will allow the relevant data to be collected without incurring the administrative overhead needed to define the exact path.

The output files can then be collected by batch file, SMS or other methods so that they can be analyzed centrally. A sample script of a routine that captures the registry key per user and posts to a central location can be found in the Appendix.

Tapping into DTrace

DTrace is the single most useful and most powerful diagnostic tool available to Enterprise Vault Administrators. In most cases, DTrace is not suited for day-to-day monitoring, and is more appropriate for

troubleshooting. However, because it is the ideal tool to capture details about problem conditions, DTrace can be an excellent part of a monitoring strategy when configured as the post-event action.

Enterprise Vault 9 SP3 introduces an enhancement to DTrace called BackTrace. BackTrace allows an Administrator to enable logging for a known condition, for example a text string associated with an error; when the string is detected, BackTrace creates a log of the process activity immediately before and during the error. More information about BackTrace can be found at <http://www.symantec.com/docs/HOWTO61102>.

In some circumstances, an Administrator may wish to collect a DTrace that starts at a certain time, or based on a specific event/circumstance, and runs forward rather than collecting what happened before the event. For example, an Administrator may want to collect detailed logging of storage-related processes when the nightly backup completes and Backup Mode flags are cleared.

DTrace will collect information from the server on which it is run; unlike other tools, it is not designed to run remotely. An Administrator can execute DTrace remotely by creating a batchfile with the DTrace syntax, setting the batchfile to run on the remote server via Windows Task Scheduler, then remotely executing the scheduled task. A sample script appears in the Appendix.

APPENDIX - Sample Scripts, Batchfiles and Syntax

Sample Log Parser Query to Tally User Upload Activity (Virtual Vault Manual Archiving Requests)

This sample can be run using the Microsoft Log Parser utility. Provide the name and path to the desired IIS log file.

```
logparser -i:iisw3c "select TO_STRING(time, 'HH') as hour, count(*) as Hits from iis_ex110215.log where cs-uri-stem like '%uploaditem.aspx%' group by Hour" -q on
```

Sample Log Parser Query to Tally User Daily Hit Count to Download.asp Page, Grouped by Hour

This sample can be run using the Microsoft Log Parser utility. Provide the name and path to the desired IIS log file.

```
logparser -i:iisw3c "select TO_STRING(time, 'HH') as hour, count(*) as Hits from iis_ex110214.log where cs-uri-stem like '%download.asp%' group by Hour" -q on
```

Script to Run DTrace Logging Via Batch File

This sample will set the StorageService, StorageCrawler and StorageFilewatch processes to Verbose logging, increase the dtrace buffer to 500,000 lines (recommended as the sample will write output to a logfile as well as the monitor which can lead to buffer overruns) and run for 1200 seconds. Edit to reflect the desired processes, log location and duration, then save with a .bat extension.

```
echo off
rem This is invoking Dtrace within a batch file.
rem Support generate the batch file, run this & then return
rem the resulting log file.
set dtlogfile=dtlog.txt
del %dtlogfile%

set dtcmds=dtracecmdfile.txt

rem Create the Dtrace command file
echo view > %dtcmds%
rem Set the individual processes per processname using the SET command and V
for Verbose
echo set StorageService v >> %dtcmds%
echo set StorageCrawler v >> %dtcmds%
echo set StorageFilewatch v >> %dtcmds%
rem Create the log file which will default to the install location where the
Dtrace utility is located using the LOG command
echo log %dtlogfile% >> %dtcmds%
rem Specify length of time to capture using the Monitor command in Seconds
```

```

echo monitor 1200 >> %dtparams%
rem Stop logging
echo log >> %dtparams%
echo yes >> %dtparams%
echo yes >> %dtparams%

rem run dtrace
dtrace 500000 <%dtparams%

rem tidy up
del %dtparams% /q

```

Note: This batchfile should be saved to the same directory as the dtrace executable, or the directory holding the dtrace executable should be added to the search path environmental variable.

Syntax to Remotely Execute a Scheduled Task

Any routine can be configured as a Windows Scheduled Task. Set the schedule to "Run Once" and configure a start date *that is in the past* to allow an Administrator to remotely execute the task on-demand or programmatically. The syntax as follows is run from the command line or can be written into a batchfile:

```
schtasks /run /s "server1.example.com" /tn "Trace.bat"
```

The /s switch signals the input for the remote server name, which should be enclosed in quotes, and the /tn indicates the configured task name.

Import File to Monitor and Alert on Disk Space Using Windows Perfmon

This sample file can be edited with environment-specific server names, drive letters, scan intervals, and batchfile name and path, then imported to enable an action-based alert whenever any monitored drive falls below 10% free space. Save the edited contents with .html extension to facilitate import via Windows Perfmon. Remove any excess counters.

```

<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;" />
<META NAME="GENERATOR" Content="Microsoft System Monitor" />
</HEAD>
<BODY>
<OBJECT ID="DISystemMonitor1" WIDTH="100%" HEIGHT="100%"
CLASSID="CLSID:C4D2D8E0-D1DD-11CE-940F-008029004347">
  <PARAM NAME="Counter00001.Path" VALUE="\SERVER1\LogicalDisk(C:)\% Free
Space"/>
  <PARAM NAME="Counter00001.AlertOverUnder" VALUE="0"/>
  <PARAM NAME="Counter00001.AlertThreshold" VALUE="10"/>

```

```

    <PARAM NAME="Counter00002.Path" VALUE="\\SERVER1\LogicalDisk(D:)\% Free
Space"/>
    <PARAM NAME="Counter00002.AlertOverUnder" VALUE="1"/>
    <PARAM NAME="Counter00002.AlertThreshold" VALUE="10"/>
    <PARAM NAME="Counter00003.Path" VALUE="\\SERVER1\LogicalDisk(E:)\% Free
Space"/>
    <PARAM NAME="Counter00003.AlertOverUnder" VALUE="1"/>
    <PARAM NAME="Counter00003.AlertThreshold" VALUE="10"/>
    <PARAM NAME="Counter00004.Path" VALUE="\\SERVER1\LogicalDisk(F:)\% Free
Space"/>
    <PARAM NAME="Counter00004.AlertOverUnder" VALUE="1"/>
    <PARAM NAME="Counter00004.AlertThreshold" VALUE="10"/>
    <PARAM NAME="Counter00005.Path" VALUE="\\SERVER1\LogicalDisk(G:)\% Free
Space"/>
    <PARAM NAME="Counter00005.AlertOverUnder" VALUE="1"/>
    <PARAM NAME="Counter00005.AlertThreshold" VALUE="10"/>
    <PARAM NAME="CounterCount" VALUE="5"/>
    <PARAM NAME="UpdateInterval" VALUE="5"/>
    <PARAM NAME="SampleIntervalUnitType" VALUE="1"/>
    <PARAM NAME="SampleIntervalValue" VALUE="5"/>
    <PARAM NAME="CommandFile" VALUE="C:\cleantemp.bat"/>
    <PARAM NAME="NetworkName" VALUE="" />
    <PARAM NAME="UserText" VALUE="" />
    <PARAM NAME="PerfLogName" VALUE="" />
    <PARAM NAME="ActionFlags" VALUE="16132"/>
    <PARAM NAME="_Version" VALUE="196611"/>
    <PARAM NAME="AlertName" VALUE="Low Disk Space Cleanup"/>
    <PARAM NAME="Comment" VALUE="This clears the temp directory when disk
space is low"/>
    <PARAM NAME="LogType" VALUE="2"/>
    <PARAM NAME="CurrentState" VALUE="1"/>
    <PARAM NAME="RealTimeDataSource" VALUE="1"/>
    <PARAM NAME="LogFileMaxSize" VALUE="-1"/>
    <PARAM NAME="DataStoreAttributes" VALUE="1"/>
    <PARAM NAME="LogFileBaseName" VALUE="Low Disk Space Cleanup"/>
    <PARAM NAME="LogFileSerialNumber" VALUE="1"/>
    <PARAM NAME="LogFileFolder" VALUE="C:\PerfLogs"/>
    <PARAM NAME="Sql Log Base Name" VALUE="SQL:!Low Disk Space Cleanup"/>
    <PARAM NAME="LogFileAutoFormat" VALUE="1"/>
    <PARAM NAME="LogFileType" VALUE="-1"/>
    <PARAM NAME="StartMode" VALUE="2"/>
    <PARAM NAME="StartAtTime" VALUE="1/31/2012 4:20:41 PM"/>
    <PARAM NAME="StopMode" VALUE="0"/>
    <PARAM NAME="RestartMode" VALUE="0"/>
    <PARAM NAME="LogFileName" VALUE="" />
</OBJECT>
</BODY>
</HTML>

```

Note: this template assumes that a file called c:\cleantemp.bat with a routine to clean the temp directory already exists on the target server. Edit the template prior to import to reflect the filename and path in your environment.

Batchfile to Clear the Temp Directory

This very simple syntax can be copied to a batchfile and run programmatically to clear the temp directory. Use a .bat extension.

```
del /q /f /s %temp%\*
```

The /q switch indicates the command should run silently (quiet) without requesting a response from the user; any temporary files that are locked will be skipped.

Powershell Script to Send Mail with Attachment

In some cases, the Administrator may wish to take the results of another operation, written to a file, and send by email. There are many approaches that can be taken. This example shows the syntax used for Windows Powershell. Copy the syntax into a text file, supplying the appropriate environment information, and save with a .ps1 attachment.

```
Add-PSSnapin Microsoft.Exchange.Management.Powershell.Admin -erroraction
silentlyContinue
$file = "C:\MyOutputFile.txt"
$smtpServer = "MyExchangeServer.example.com"
$att = new-object Net.Mail.Attachment($file)
$msg = new-object Net.Mail.MailMessage
$smtp = new-object Net.Mail.SmtpClient($smtpServer)
$msg.From = "monitoring.address@example.com"
$msg.To.Add("EVAdmin.Team@example.com")
$msg.Subject = "Monitoring Report"
$msg.Body = "Attached is the Monitoring report"
$msg.Attachments.Add($att)
$smtp.Send($msg)
$att.Dispose()
```

The .ps1 file can be executed from within the Powershell shell, or from command line by drilling down to the directory holding the Powershell executable:

```
C:\windows\system32\windowspowershell\v1.0\powershell.exe c:\MyMailer1.ps1
```

SQL Script to Total Hourly Archiving Rate Across Multiple Servers

The following SQL script can be run on demand or via a schedule and will calculate the hourly archiving rate for one or more Enterprise Vault servers. This script can be extremely helpful in detecting performance issues, especially in a Journal Archiving environment where archiving generally occurs 24 hours a day.

```
USE EnterpriseVaultDirectory
```

```

SELECT ArchivedDate, SUM(NumberOfItems)

FROM

(
    SELECT LEFT(convert(varchar, s.archiveddate,20),13) AS
ArchivedDate,

COUNT(s.ArchivedDate) AS NumberOfItems

FROM EVJournalVault01.dbo.saveset s,
EVJournalVault01.dbo.savesetproperty sp

WHERE s.savesetidentity = sp.savesetidentity

AND s.archiveddate > getdate() - 3

GROUP BY LEFT(CONVERT(varchar, archiveddate,20),13)

UNION

SELECT LEFT(convert(varchar, s.archiveddate,20),13) AS
ArchivedDate,

COUNT(s.ArchivedDate) AS NumberOfItems

FROM EVJournalVault02.dbo.saveset s,
EVJournalVault012010.dbo.savesetproperty sp

WHERE s.savesetidentity = sp.savesetidentity

AND s.archiveddate > getdate() - 3

GROUP BY LEFT(CONVERT(varchar, archiveddate,20),13) ) Q1

GROUP BY ArchivedDate

ORDER BY ArchivedDate desc

```

This nested query can be tailored to the environment as follows:

- Repeat the *italicized* section beginning with the second iteration of “SELECT” and ending just before the UNION, once for each Vault Store database to be queried
- Replace the bolded reference to the database name to match the database names in use in the environment; the *.dbo.tablename* string should remain intact
- Include the UNION instruction between each query sub-section. A UNION is not required after the final iteration
- If the Vault Store databases reside on a different SQL server than the Directory database, then the SQL servers may need to be configured as Linked Servers, and the database name string

may need to be appended with the SQL server name. These details will vary with the environment.

SQL Script to Check for Users With Persistent Vault Cache Sync Failures

This SQL script will identify users who have Vault Cache sync failures with no recorded successful job. This script will not identify a user who has failed after a successful sync, or who have been successful after a failure, but will identify those who have never been successful. Identifying consistently unsuccessful users allows the Administrator to proactively assist end users.

```
USE EnterpriseVaultDirectory
SELECT jobAuthor, jobDescription, jobState, retryCount, startTime, endTime
FROM AdminJobEntry WHERE jobType = 8
    AND jobState = 6
    AND jobAuthor NOT IN (SELECT jobAuthor FROM AdminJobEntry WHERE jobType = 8
AND jobState != 1)
```

The query can be refined by adding a condition based on the StartTime (for example, only show results where the job's start time is greater than one week).

SQL Script to Monitor MoveArchive Statistics

This script can be executed as a SQL scheduled job with the results emailed to the Administrator periodically, or can be run on-demand or called by a homegrown web page, and will show details of MoveArchive jobs. The query can be refined by the start time or total elapsed time, the stage of the job, and so forth, to focus attention on specific conditions or routines.

```
USE EnterpriseVaultDirectory
SELECT AH.VaultName "Archive Name",
    'Items To Move' = SUM(AMH.CountItemsToMove - AMH.CountItemsMoved),
    AMH.CountItemsFailed "Items Failed",
    AH.VaultStoreName "Source Vault Store",
    AMH.DestinationVaultStoreName "Destination Vault Store",
    'Progress' = CASE WHEN Substring(S.Progress, 14,1) = 1 THEN 'Copying
Items' WHEN Substring(S.Progress, 14,1) = 2 THEN 'Waiting To Update
Shortcuts' WHEN Substring(S.Progress, 14,1) = 3 THEN 'Updating Shortcuts'
WHEN Substring(S.Progress, 14,1) = 4 THEN 'Step 4' WHEN Substring(S.Progress,
14,1) = 5 THEN 'Verifying Items' ELSE 'Unknown' END,
    'Status' = CASE WHEN S.State = 1 THEN 'In Progress' WHEN S.State = 6
THEN 'Errored' WHEN S.State = 10 THEN 'Completed' WHEN S.State = 11 THEN
'Completed With Errors' WHEN S.State = 12 THEN 'Queued' WHEN S.State = 13
THEN 'Waiting' ELSE 'Unknown' END,
    'Hours In Progress' = DATEDIFF(hh, AH.startTimeUTC, GETDATE())
FROM SubTask S,
    ArchiveHistory AH,
    ArchiveMoveHistory AMH
```

```

WHERE substring(s.Parameters, 47, 56) = AH.VaultEntryId
  AND AH.ArchiveHistoryIdentity = AMH.ArchiveHistoryIdentity
  AND AH.OperationType = 0
  AND S.State NOT IN (6,9,10,11)
GROUP BY AH.VaultName, AMH.CountItemsFailed, AH.VaultStoreName,
AMH.DestinationVaultStoreName, S.Progress, S.State, AH.StartTimeUTC ORDER BY
S.State, Progress ASC

```

SQL Statement to List Youngest (Newest) Indexed Item per Archive

This sample will list the most recently indexed item in each archive, which can validate proper operating by verifying that archives have recently had data added, and that the data has been successfully indexed. It is normal from time to time for user mailbox archives to not receive new data, such as when a user is on vacation and does not generate or receive any mail; analyze not only which archives are behind but by how much before concluding there is evidence of a problem.

```

Use enterprisevaultdirectory
select
archivename, foldername as IndexName
, youngestitemdateutc
, youngestarchiveddateutc
from root
inner join archive on root.rootidentity = archive.rootidentity
inner join indexvolume on archive.rootidentity = indexvolume.rootidentity

```

SQL Script to Identify Archives With Items Missing From the Index

In environments with very strict e-discovery requirements, it may be important to be alerted when the Indexing service cannot index a given item. In most of these cases, an unindexable item will in and of itself not cause the index to fail, and an event will be written to the Event Logs of the Enterprise Vault server hosting the indexing service, but a centralized accounting may be more useful. This script can be run on-demand or as part of a scheduled job and will give details on those indexes containing failed items, such as the archive name, storage location and index status.

```

SELECT  A.ArchiveName "Archive Name",
CE.ComputerName "Vault Store Server", VSE.VaultStoreName "Vault Store",
(IRP.IndexRootPath + '\' + IV.FolderName) "Index Folder",
IV.IndexedItems "Indexed Items", IV.OldestItemDateUTC "Oldest Item (Sent)",
IV.YoungestItemDateUTC "Newest Item (Sent)",
IV.FailedItems "Failed Items", IV.Offline "Index Offline", IV.Failed "Index
Failed", IV.Rebuilding "Rebuilding"
FROM EnterpriseVaultDirectory.dbo.Archive A,
     EnterpriseVaultDirectory.dbo.Root R,
     EnterpriseVaultDirectory.dbo.VaultStoreEntry VSE,
     EnterpriseVaultDirectory.dbo.StorageServiceEntry SSE,
     EnterpriseVaultDirectory.dbo.ComputerEntry CE,
     EnterpriseVaultDirectory.dbo.IndexRootPathEntry IRP,

```

```

EnterpriseVaultDirectory.dbo.IndexVolume IV
WHERE A.RootIdentity = R.RootIdentity
      AND VSE.StorageServiceEntryId = SSE.ServiceEntryId
      AND SSE.ComputerEntryID = CE.ComputerEntryID
      AND A.VaultStoreEntryID = VSE.VaultStoreEntryID
      AND R.RootIdentity = IV.RootIdentity
      AND IV.IndexRootPathEntryID = IRP.IndexRootPathEntryID
      AND IV.FailedItems != 0

```

SQL Script to Retrieve Task Status from the EnterpriseVaultMonitoring Database

When installed, the Enterprise Vault Monitoring Agent runs on each Enterprise Vault server and periodically writes status information to the EnterpriseVaultMonitoring database. Query this central repository for Task status rather than writing complex scripts:

```

SELECT CE.ComputerName "EV Server",
       T.Name "Task Name",
       CASE TM.Status
         WHEN 0 THEN 'Stopped'
         WHEN 1 THEN 'Running'
         WHEN 2 THEN 'Paused'
         WHEN 3 THEN 'Disabled'
         WHEN 4 THEN 'Loading'
         WHEN 5 THEN 'Stopping'
         WHEN 6 THEN 'Failed'
         WHEN 7 THEN 'Refreshing'
         WHEN 8 THEN 'Service Unavailable'
         WHEN 9 THEN 'Error'
         WHEN 10 THEN 'Completed'
         WHEN 11 THEN 'Completed With Errors'
         WHEN 12 THEN 'Queued'
         WHEN 13 THEN 'Not In Schedule'
       END AS "Task Status",
       TM.EntryTime "Time Checked"
FROM   EnterpriseVaultMonitoring.dbo.TaskMonitoring TM,
       EnterpriseVaultDirectory.dbo.Task T,
       EnterpriseVaultDirectory.dbo.ComputerEntry CE WHERE  TM.TaskID =
T.TaskEntryID
      AND T.ComputerEntryId = CE.ComputerEntryId ORDER BY CE.ComputerName,
T.Name

```

SQL Script to Tally Users Enabled For Archiving in Last 24 Hours

This script should be run against each Vault Store database; combine results from multiple databases using a UNION clause. Note that the archive's create date is stored in UTC; when establishing a schedule for this script, reflect the local timezone offset for most accurate results.

```
USE EVExchangeVaultStore
Select count (createdDate)
FROM ArchivePoint
WHERE CreatedDate > (GETDATE() - 1)
```

SQL Script to Send SQL DB_Mail With Prior Day's Archiving Numbers

This SQL Script will run a query against the Vault Store database, collect the prior day's archiving totals, and send via SQL Database Mail. To include additional Vault Store databases in the totals, the Administrator can either configure additional jobs, or add additional queries to the same step of the existing job, using a UNION clause.

This script requires a SQL Mail Profile which can be configured through SQL Management Studio → Management → Database Mail. Configure the @profile_name, @recipients, and other details as appropriate for the environment. Any query can be placed into the @query line, taking care to include the correct database name.

It is essential that this routine be run in the context of the MSDB; the routine will fail when run on any other database context. This script should be scheduled to run after the archiving run has completed and is based off of the current date minus one calendar day; adjust the GETDATE value or increment if needed.

```
USE MSDB
EXEC sp_send_dbmail
    @profile_name = 'EVMailer',
    @recipients = 'AdminTeam@example.com',
    @subject = 'Nightly Archiving Rate',
    @body = 'Last night numbers are as follows',
    @execute_query_database = 'EVExchangeVaultStore',
    @query = ' USE EVExchangeVaultStore Select  CAST(SUM(Itemsize/1024) AS
Bigint ) as Total_MB,
    Count_Big (idtransaction) as ItemCount
    From Saveset Where ArchivedDate < (GETDATE() - 1) '
```

Batch File to Collect EV Client/Vault Cache Registry Keys and Write to a Central Location

This batchfile can be assigned to users or to machines as a login script and will collect the contents of HKEY_CURRENT_USER\Software\KVS\Enterprise Vault\Client. The results can then be centrally analyzed by an Administrator. Replace the [\\AdminServer\EVLogs](#) with a share that exists on the network, and select an appropriate letter drive mapping. Save with a .bat extension.

Note: end users will need a minimum of Write access to the central location, or the login script should be configured to run under an appropriate user context.

Additional criteria can be added; for example, include the time/date stamp in the name of the file to run the file on each login, generating additional logs each day, or include a step to suppress the batchfile if the user's output log already exists.

```
NET USE P: \\AdminServer\EVLogs
CD\
CD OutputLogs
DEL *.*
CD\
REG EXPORT "HKEY_CURRENT_USER\Software\KVS\Enterprise Vault\Client"
c:\OutputLogs\EVClientReg.txt
RENAME c:\OutputLogs\EVClientReg.txt %Username%EVClientReg.txt
COPY *.* p:
NET USE \DELETE P:
```

Visual Basic Script to Consolidate Client Registry Files

This script will extract four registry keys from a .txt or .reg export file: MDC Last Attempt, MDC Last Success, MDC Next Sync and Content Last Sync. These keys can be crucial in evaluating the health of end user Vault Caches.

This example assumes that the files already exist in a central location such as a fileshare, and that each file has a unique name; a technique to gather registry files in this manner is shown in the sample above. This example also assumes that the installed client version supports these registry keys; the keys targeted in this example should be tailored to your installed client version. Finally, this script will replace any existing output from a previous run; include an instruction to rename older iterations (either in the VB itself or in an overlying batch file) if you wish to maintain a history.

Update the reportFile and regLocation values to reflect your desired input and output locations, then save with a .vbs extension. To run the script, use the following syntax from the command line: cscript parseRegistry.vbs (where parseRegistry.vbs is the filename given when saving).

```

Option Explicit
'On Error Resume Next

Dim reportFile, MDCLastGoodSync, MDCLastSync, MDCNextSync, CCLastSync
Dim regLocation, regFile, objFSO, objFolder, objReadFile, regFileList,
textLine, folderIdx, i, startTime, endTime

const ForReading = 1

i = 0
startTime = Now
reportFile = "C:\OutputReport.csv"
regLocation = "C:\Test\"

Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFolder = objFSO.GetFolder(regLocation)
Set regFileList = objFolder.Files

deleteReport(reportFile)
createReport reportFile

Wscript.echo now() & ": Starting Scan"
Wscript.echo now() & ": Scanning the following folder: " & regLocation
WScript.echo now() & ": Report will be written to: " & reportFile

For Each folderIdx in regFileList
    regFile = regLocation & folderIdx.Name
    Wscript.echo now() & ": Reading " & regFile
    Set objReadFile = objFSO.OpenTextFile(regFile, ForReading, False, -1)

    Do Until objReadFile.AtEndOfStream
        textLine = objReadFile.ReadLine
        If InStr(1, textLine, "OVMDCLastSyncTime") <> 0 Then
            MDCLastSync = getMDCLastSync(textLine)
        ElseIf InStr(1, textLine, "OVMDCNextSyncTime") <> 0 Then
            MDCNextSync = getMDCNextSync(textLine)
        ElseIf InStr(1, textLine, "OVMDCLastGoodSyncTime") <> 0 Then
            MDCLastGoodSync = getMDCLastGoodSync(textLine)
        ElseIf InStr(1, textLine, "OVCCLastCompletedTime") <> 0 Then
            CCLastSync = getCCLastCompleted(textLine)
        End If
    Loop

    objReadFile.Close

    addInfo folderIdx.Name, MDCLastGoodSync, MDCLastSync, MDCNextSync,
CCLastSync, reportFile
    i = i + 1
Next

endTime = Now
Wscript.echo now() & ": Scanned " & i & " files in " & DateDiff("s",
startTime, endTime) & " seconds"
Wscript.echo now() & ": Completed Scan"

```

```

Public Function getMDCLastSync(lastSync)
    Dim Year, Month, Day, Time
    Year = MID(lastSync, 22, 4)
    Month = MID(lastSync, 27, 2)
    Day = MID(lastSync, 30, 2)
    Time = MID(lastSync, 33, 8)
    getMDCLastSync = Month & "/" & Day & "/" & Year & " " & Time
End Function

Public Function getMDCNextSync(nextSync)
    Dim Year, Month, Day, Time
    Year = MID(nextSync, 22, 4)
    Month = MID(nextSync, 27, 2)
    Day = MID(nextSync, 30, 2)
    Time = MID(nextSync, 33, 8)
    getMDCNextSync = Month & "/" & Day & "/" & Year & " " & Time
End Function

Public Function getMDCLastGoodSync(lastGood)
    Dim Year, Month, Day, Time
    Year = MID(lastGood, 26, 4)
    Month = MID(lastGood, 31, 2)
    Day = MID(lastGood, 34, 2)
    Time = MID(lastGood, 37, 8)
    getMDCLastGoodSync = Month & "/" & Day & "/" & Year & " " & Time
End Function

Public Function getCCLastCompleted(lastCompleted)
    Dim Year, Month, Day, Time
    Year = MID(lastCompleted, 26, 4)
    Month = MID(lastCompleted, 31, 2)
    Day = MID(lastCompleted, 34, 2)
    Time = MID(lastCompleted, 37, 8)
    getCCLastCompleted = Month & "/" & Day & "/" & Year & " " & Time
End Function

sub deleteReport(reportFile)
    Dim objFSO
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    If objFSO.FileExists(reportFile) Then
        objFSO.DeleteFile reportFile
    End If
End Sub

sub createReport(reportFile)
    Dim objFSO,objTextFile
    set objFSO = createobject("Scripting.FileSystemObject")
    set objTextFile = objFSO.CreateTextFile(reportFile)
    objTextFile.WriteLine("Username, MDC Last Attempt, MDC Last Success,
MDC Next Sync, Content Last Sync")
    objTextFile.Close
    SET objTextFile = nothing

```

End sub

```
sub addInfo(fileName, MDCLastGoodSync, MDCLastSync, MDCNextSync, CCLastSync,
reportFile)
```

```
    Dim objFSO,objTextFile
    set objFSO = createobject("Scripting.FileSystemObject")
    set objTextFile = objFSO.OpenTextFile(reportFile, 8, True)
    objTextFile.WriteLine(fileName & "," & MDCLastSync & "," &
MDCLastGoodSync & "," & MDCNextSync & "," & CCLastSync)
    objTextFile.Close
    set objTextFile = nothing
```

End Sub

About Symantec:

Symantec is a global leader in providing storage, security and systems management solutions to help consumers and organizations secure and manage their information-driven world.

Our software and services protect against more risks at more points, more completely and efficiently, enabling confidence wherever information is used or stored.

For specific country offices and contact numbers, please visit our Web site: www.symantec.com

Symantec Corporation
World Headquarters
350 Ellis Street
Mountain View, CA 94043 USA
+1 (650) 527 8000
+1 (800) 721 3934

Copyright © 2011 Symantec Corporation. All rights reserved. Symantec and the Symantec logo are trademarks or registered trademarks of Symantec Corporation or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.