VMSSPI for VSI OpenVMS

User Guide

Publication Date: June 2025

Operating Systems: VSI OpenVMS Alpha Version 8.4-2L1 or higher
VSI OpenVMS IA-64 Version 8.4-2L1 or higher
VSI OpenVMS x86-64 Version 9.2-2 or higher

Software Version: VMSSPI V9.0-27

Kit Names: VSI-AXPVMS-VMSSPI-V0900-27-1.PCSI
VSI-I64VMS-VMSSPI-V0900-27-1.PCSI
VSI-X86VMS-VMSSPI-V0900-27-1.PCSI

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

VMSSPI for VSI OpenVMS User Guide

Copyright © 2025 VMS Software, Inc. (VSI), Boston, Massachusetts, USA

Legal Notice

Confidential computer software. Valid license from VSI required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

The information contained herein is subject to change without notice. The only warranties for VSI products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. VSI shall not be liable for technical or editorial errors or omissions contained herein.

All other trademarks and registered trademarks mentioned in this document are the property of their respective holders.

Table of Contents

1. Overview 6
2. Changes Included in This Release 6
3. VMSSPI Functionality 6
    3.1. The SYSTEM Module 6
    3.2. The PERFORMANCE Module 8
    3.3. The SECURITY Module 10
    3.4. The METRICS Module 11
4. Installation 12
    4.1. Requirements 12
    4.2. Installing the Kit 13
5. Post-Installation Tasks 14
6. VMSSPI Logical Names 16
    6.1. VMSSPI$DATA 16
    6.2. VMSSPI$NO_CLUSTER_CHECKS 16
    6.3. VMSSPI$IGNORE 16
    6.4. VMSSPI$RERAISE_INTERVAL 17
    6.5. TCPIP$SMTP_FROM 17
7. VMSSPI Configuration File 18
    7.1. Generating the Configuration File 18
    7.2. Reconfiguring 19
    7.3. Editing the Configuration File 19
        7.3.1. Entering the Cluster Name Information 20
        7.3.2. Defining the Time Interval Between Two Consecutive Checks 20
        7.3.3. Defining Periods to Restrict Monitoring 20
        7.3.4. Defining a Process to Monitor 21
        7.3.5. Defining Disks to Monitor 22
        7.3.6. Specifying Batch Queues 24
        7.3.7. Specifying Print Queues 25
        7.3.8. Specifying Batch Jobs 26
        7.3.9. Specifying Shadow Sets 27
        7.3.10. Specifying a Rules File Using INCLUDE 27
        7.3.11. Enabling Intrusion Detection 28
        7.3.12. LAN Devices to Monitor 28
        7.3.13. Security Filter Setting 28
8. Log Files 29
9. The Messages File 29
    9.1. Defined Messages 35
        9.1.1. VMSSPI_AUDSRV_Audit 35
        9.1.2. VMSSPI_AUDSRV_AuditServer 35
        9.1.3. VMSSPI_AUDSRV_Breakin 36
        9.1.4. VMSSPI_AUDSRV_Connection 36
        9.1.5. VMSSPI_AUDSRV_Install 36
        9.1.6. VMSSPI_AUDSRV_Logfailure 36
        9.1.7. VMSSPI_AUDSRV_Login 37
        9.1.8. VMSSPI_AUDSRV_Logout 37
        9.1.9. VMSSPI_AUDSRV_Mount 37
        9.1.10. VMSSPI_AUDSRV_NCP 37
        9.1.11. VMSSPI_AUDSRV_Netproxy 38
        9.1.12. VMSSPI_AUDSRV_ObjectAccess 38
        9.1.13. VMSSPI_AUDSRV_ObjectCreate 38

9.1.14. VMSSPI_AUDSRV_ObjectDelete 38
9.1.15. VMSSPI_AUDSRV_Process 39
9.1.16. VMSSPI_AUDSRV_Rightslist 39
9.1.17. VMSSPI_AUDSRV_Sysgen 39
9.1.18. VMSSPI_AUDSRV_Systime 39
9.1.19. VMSSPI_AUDSRV_Sysuaf 40
9.1.20. VMSSPI_ActiveCPU 40
9.1.21. VMSSPI_BatchQueue 40
9.1.22. VMSSPI_ClusterMemberAdded 40
9.1.23. VMSSPI_ClusterMemberRemoved 41
9.1.24. VMSSPI_DiskState 41
9.1.25. VMSSPI_FreeSpace 42
9.1.26. VMSSPI_HardwareError 42
9.1.27. VMSSPI_Intruder 42
9.1.28. VMSSPI_LANCarrierCheckFailures 42
9.1.29. VMSSPI_MemberState 43
9.1.30. VMSSPI_NPAGEDYN 43
9.1.31. VMSSPI_PendingJobs 43
9.1.32. VMSSPI_PrintQueue 44
9.1.33. VMSSPI_ProcessLooping 44
9.1.34. VMSSPI_ProcessOccurrences 44
9.1.35. VMSSPI_ProcessState 45
9.1.36. VMSSPI_QueueManager 45
9.1.37. VMSSPI_Quota_ASTLM 45
9.1.38. VMSSPI_Quota_BIOLM 45
9.1.39. VMSSPI_Quota_BYTLM 46
9.1.40. VMSSPI_Quota_DIOLM 46
9.1.41. VMSSPI_Quota_ENQLM 46
9.1.42. VMSSPI_Quota_FILLM 46
9.1.43. VMSSPI_Quota_PGFLQUOTA 47
9.1.44. VMSSPI_Quota_PRCLM 47
9.1.45. VMSSPI_Quota_TQELM 47
9.1.46. VMSSPI_RetainedJobs 47
9.1.47. VMSSPI_SecurityServer 48
9.1.48. VMSSPI_ShadowSetState 48
9.1.49. VMSSPI_Started 48
9.1.50. VMSSPI_BatchJobMissing 48
9.1.51. VMSSPI_BatchJobOverrun 49
9.1.52. VMSSPI_BatchQueueNotStarted 49
9.1.53. VMSSPI_BatchQueueNotStopped 49
9.1.54. VMSSPI_CPUutilization 49
9.1.55. VMSSPI_DiskNotAvailable 50
9.1.56. VMSSPI_DiskQueueLength 50
9.1.57. VMSSPI_DiskWriteLocked 51
9.1.58. VMSSPI_Hotfile 51
9.1.59. VMSSPI_LANutilization 51
9.1.60. VMSSPI_MemoryUtilization 52
9.1.61. VMSSPI_MissingMember 52
9.1.62. VMSSPI_PageFileFreeSpace 53
9.1.63. VMSSPI_ProcessMissing 53
9.1.64. VMSSPI_ProcessSlots 53
9.1.65. VMSSPI_ReshashtblDense 54

9.1.66. VMSSPI_ReshashtblSparse 54
9.1.67. VMSSPI_ShadowSetMissing 55
9.1.68. VMSSPI_SwapFileFreeSpace 55
9.1.69. VMSSPI_SystemBIOrate 56
9.1.70. VMSSPI_SystemDIOrate 56
9.1.71. VMSSPI_SystemPageFaultRate 57
10. Alerting Module Configuration 57
10.1. DataDog 57
10.2. Dynatrace 58
10.3. The File Module 59
10.4. MQTT Interface Module 59
10.5. The NULL Module 60
10.6. PagerDuty 61
10.7. Slack Integration 61
10.8. Splunk 62
10.9. Syslog Interface 62
10.10. RabbitMQ Interface Module 64

1. Overview

VSI VMSSPI for VSI OpenVMS is a software package that can be used to monitor for and report on system-, performance-, and security-related events on OpenVMS systems. VMSSPI includes comprehensive coverage of such events and provides interfaces to popular enterprise monitoring and alerting services such as DataDog, Splunk, PagerDuty, and Dynatrace in addition to providing facilities to report events via email, to a remote Linux syslog daemon, or to a Slack channel.

2. Changes Included in This Release

- Documentation relating to configuration of the experimental METRICS module, which makes various performance metrics used by VMSSPI available to Prometheus (or any other tool that supports the Prometheus protocol).
- Added functionality to the metrics interface to report percent CPU utilization per active CPU.
- Added functionality to the metrics module to allow the default port number used by the metrics interface to be overridden by the value defined via the logical name VMSSPI$METRICS_PORT.
- This release includes a new alerting module (VMSSPI$RABBITMQ_SHR.EXE) that can be used to publish alert messages to a RabbitMQ topic exchange.

3. VMSSPI Functionality

VMSSPI consists of the following modules:

Module	Description
SYSTEM	Reports on a number of system-related items such as processes, disks, shadow sets, queues, etc.
PERFORMANCE	Reports on system and process resource utilization.
SECURITY	Reports in real-time on security events that the AUDIT_SERVER process detects.
METRICS	Makes various performance metrics used by other VMSSPI modules available to the Prometheus monitoring solution.

The work performed in these modules is described in the sub-sections below.

3.1. The SYSTEM Module

The SYSTEM module is activated at regular time intervals (usually, every minute) and reports on the items summarised in the following list:

- Changes in the error count

  Checks at regular time intervals if the error count on CPU, memory, and devices has changed. Note that upon start-up of this module, no messages are sent regarding the current error count on devices. However, reporting is always done on the CPU and memory error count.

- OpenVMS cluster changes

Sends a notification whenever a node is added to or removed from an OpenVMS cluster.

- Process availability

  Reports on the availability of processes. A process is defined by its OpenVMS process name and UIC.

  Note the following points:

  - Specification of the UIC is optional. If you do specify the UIC, use the identifier format or numeric format.
  - The process name can contain the asterisk (*) and/or the percent sign (%) wildcard.
  - You can also check for a minimum number of occurrences if the process name contains the asterisk (*) and/or the percent sign (%) wildcard.
  - Process availability can be checked clusterwide.
  - You can also restrict process availability checks during predefined periods.

- Disks

  Reports on disks status and available free space. A disk is specified by its physical name or its logical volume name. For each disk, the SYSTEM module determines the following:

  - What is the status of the disk? For example, is the disk correctly mounted and accessible?
  - Is the disk write-locked?
  - Is there free space on the disk?

  Monitoring of free space on clusterwide mounted disks is done on a clusterwide basis. Disk state and write lock monitoring are done on a per-node basis.

  Monitoring of a disk can be restricted to specific periods of time.

- Queue manager status

  Checks the status of the queue manager or managers and reports if the status is different from "Running".

- Batch queue status

  Reports on the status of batch queues and the number of retained and pending jobs on batch queues.

  The preferred status of a batch queue can be either "Started" (indicated by a status of "Idle", "Available", or "Busy") or "Stopped" during different time periods. The SYSTEM module checks that this preferred status corresponds to the actual status of the batch queue.

  Batch queue monitoring is usually performed on a clusterwide basis.

- Print queue status

  Reports on the status of print queues and the number of pending and retained jobs on print queues. Print queue monitoring is usually performed on a clusterwide basis.

- Batch jobs

  Reports on the availability of batch jobs. A batch job is defined by its job name, user name, and batch queue name. Corollaries to this definition are the following:

  - A job name can contain wildcards.
  - A batch queue can be a generic queue or an execution queue.
  - If a generic queue has been specified and the job is currently running on one of the execution queues, the job is considered to be present.

  Batch job monitoring (which is usually performed clusterwide) can be restricted to certain time periods.

- Shadow sets

  Reports on the availability of shadow sets and the status of their members, including missing members, unexpected members, and copy/merge operations. Monitoring can be restricted to certain time periods.

  Clusterwide mounted shadow sets are usually monitored on a clusterwide basis.

- System intruders

  Reports on system intruders.

3.2. The PERFORMANCE Module

The function of the PERFORMANCE module is to monitor resource usage and system/process performance. More specifically, the module reports on the following parameters:

- CPU utilization

  The value reported is the actual CPU utilization over the total CPU capacity times 100. This value is calculated over a period of 10 minutes.

  All CPUs fully used equals 100%.

- Memory utilization

  The value reported is the average percentage of memory utilization over a period of 5 minutes.

- Page file utilization

  Reports on the free space in the system page file or files. The value reported is the percentage of free space in the page file or files.

- Swap file utilization

  Reports on the free space in the system swap file or files. The value reported is the percentage of free space in the swap file or files.

- Buffered I/O count

  Reports on the total number of buffered I/Os per second, for the entire system. The value reported is the average BIO rate over a period of 5 minutes.

- Direct I/O count

  Reports on the total number of direct I/Os per second, for the entire system. The value reported is the average DIO rate over a period of 5 minutes.

- Processes in COM or COMO state

  Reports on the average number of processes in the COM or COMO state over a period of 5 minutes.

- Total number of processes

  Reports on the current number of processes on the system. The value reported is the average number of processes over the value of the SYSGEN parameter MAXPROCESSCNT times 100. This value is calculated over a period of 5 minutes.

- Non-paged pool expansions

  Reports on the number of expansions of the non-paged dynamic memory and on the number of times the system failed to expand non-paged dynamic memory.

- System page faults

  Reports on the average number of system page faults per second, over a period of five minutes.

- Resource hash table utilization

  Reports on the total number of resources on the system, compared to the value of the SYSGEN parameter RESHASHTBL times 100.

  The average utilization is calculated over a period of 5 minutes.

- LAN device utilization

  Reports on the Ethernet device throughput of selected LAN devices. The value reported is the line utilization to the line speed times 100 for the last minute.

- Looping processes

  Continuously looks for processes that use a lot of CPU but do not do any I/O. A process is flagged as "seems to be looping" if it uses a minimum of 25% of one CPU and does not do any I/O during a period of at least 2 minutes.

- Active CPUs

  Sends a notification message when it detects a stopped CPU on the system.

- Processes in special states

  Sends a notification if processes are in a state other than LEF, LEFO, CEF, HIB, HIBO, COM, or CUR. The PERFORMANCE module takes multiple samples of the state of each process, and notification is sent only if a process is in the same special state for a minimum of 1 minute.

- Disk I/O

  Reports on the following items for selected disks:

  - The DIO rate on the disk. The value reported is the average number of DIOs made to this disk per second, over a time of 10 minutes.

- The queue length on the disk. The value reported is the average queue length per second on this disk, over a time of 10 minutes.
- Hot files on the disk. The PERFORMANCE module reports on the files on which the highest number of DIOs per second are made.

- Process quota utilization

  Reports on the quota utilization of each process on the system. A message is sent when a process is detected whose utilization of one of its quotas reaches its limit.

  The PERFORMANCE module checks on the following quotas:

  - ASTLM
  - BIOLM
  - DIOLM
  - BYTLM
  - ENQLM
  - FILLM
  - PGFLQUOTA
  - PRCLM
  - TQELM

  Note that in some situations, such as for very short-lived processes or for processes that are starting or terminating at the time of a VMSSPI process scan, anomalous alerts may occasionally be reported.

3.3. The SECURITY Module

The AUDIT_SERVER process records security-relevant activity as it occurs on the system, that is, any activity related to user access to the system or a protected object within the system, including the following:

- Logins, logouts, and login failures
- Changes to the user authorization, rights list, and network proxy files
- Access to protected objects such as files, devices, global sections, queues, etc
- Changes to the security attributes of protected objects

The AUDIT_SERVER usually writes security event messages to the following locations:

- Security audit log file

  Security events are written in binary format to this clusterwide log file. Each record contains details related to the event and can be examined using the ANALYZE/AUDIT utility. The **`SET AUDIT/AUDIT`** command allows you to specify which security events are to be written to the

audit log file (see help for the **SET AUDIT** command for additional information on configuring the audit logging facility and which events will be reported).

- Security operator terminals

  These are terminals that are enabled to receive OPCOM security class messages, allowing the system manager to monitor user activity in real time. The **SET AUDIT/ALARM** command allows you to specify which security events are to be written to operator terminals.

The VMSSPI SECURITY module allows you to monitor security events in a continuous but simplified manner. System managers can quickly detect suspicious security events, which they can then examine further using the ANALYZE/AUDIT utility.

The operator terminal and the security audit log file are the primary destinations for security event messages. An additional feature of the security auditing facility is a listener mailbox, which you can create to receive a binary copy of all security-auditing messages.

The SECURITY module does the following:

1. Creates and reads the listener mailbox
2. Processes the auditing information. The AUDIT_SERVER writes all monitored security items to the listener mailbox; this information is summarized in a one-line message.
3. Sends a message to the configured VMSSPI consumer module (and potentially, via email)

The messages written to the listener mailbox are the ones that result from entering the **SET AUDIT/ALARM** and/or **SET AUDIT/AUDIT** commands. VSI recommends that certain security events be enabled for auditing, for example:

```
$ SET AUDIT/AUDIT/ENABLE=(ACL, -
             AUTHORIZATION, -
             BREAKIN:(ALL), -
             LOGFAILURE:(ALL), -
             FILE_ACCESS:FAILURE:(READ,WRITE,EXECUTE,DELETE,CONTROL), -
             LOGIN:(DIALUP))
```

Use the **SHOW AUDIT** command to check the security events that are enabled for monitoring on your system.

3.4. The METRICS Module

The VMSSPI METRICS module (VMSSPI$METRICS.EXE) is started on VMSSPI start-up if the logical name VMSSPI$EXPORT_METRICS is defined (to anything) at the system level. When this facility is enabled, various system-level and per-process performance metrics calculated by other VMSSPI modules are sent to the VMSSPI$METRICS process, making the metrics available for consumption by the Prometheus monitoring solution (https://prometheus.io/). Note that this module is currently experimental and will be documented in greater detail for future releases of VMSSPI; if you wish to make use of this module, Prometheus should be configured to access these metrics at the TCP/IP port 8000 on the OpenVMS server in question. The logical name VMSSPI$METRICS_PORT can be used to define an alternative port number.

The system metrics currently exposed are as follows:

- Average CPU utilization (averaged across all active CPUs)

- Percent CPU utilization per active CPU
- Page file free space
- Swap file free space
- Overall memory utilization
- System BIO rate
- System DIO rate
- System page fault rate
- Bytes sent (per network interface)
- Bytes received (per network interface)
- DIO rate per monitored disk
- I/O queue length per monitored disk

In addition to the above system metrics, disk space used (expressed as a percentage) will be reported on a per-disk basis for any disks specified for monitoring in the VMSSPI configuration file for which the **/METRICS** qualifier is specified. Note that if disks are monitored for specific periods, metrics will only be made available to Prometheus during those periods.

The **/METRICS** qualifier can also be included for any processes specified for monitoring in the VMSSPI configuration file, whereupon the following metrics will be made available (on a per-process basis) to the Prometheus interface, with the reported values representing the change over the past one-minute interval.

- CPU time consumed by the process (across all modes)
- Page faults
- Direct I/O operations
- Buffered I/O operations

Note that the VMSSPI$METRICS process implements an embedded web server, and that HTTP only is supported by the current version. The web server does not access any file systems and has access to no data other than the metrics data supplied to it. Support for HTTPS will be provided in a future release.

4. Installation

This section describes how to install the VMSSPI product. Note that if you are currently using the old HPE VMSSPI product in conjunction with HPE OpenView, this old product must be fully removed before installing VSI VMSSPI for VSI OpenVMS.

4.1. Requirements

In order to install and use VMSSPI on VSI OpenVMS Alpha, IA-64, and x86-64, the following software products and versions are required:

- OpenVMS V8.4-2L1 or higher (Alpha), OpenVMS 8.4-2L1 or higher (IA-64), OpenVMS 9.2-2 or higher (x86-64)
- VSI TCP/IP

In addition to these requirements, installing VSI VMSSPI for VSI OpenVMS on an ODS-5 formatted file system is recommended.

4.2. Installing the Kit

The kit is provided as an OpenVMS PCSI kit that can be installed by a suitably privileged user using the following command:

```
$ PRODUCT INSTALL VMSSPI
```

The installation will then proceed as follows. Note that output may differ slightly from that shown depending on platform, product version, and various other factors:

```
Performing product kit validation of signed kits ...

The following product has been selected:
    VSI AXPVMS VMSSPI V9.0-27               Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and
for any products that may be installed to satisfy software dependency
requirements.

Configuring VSI AXPVMS VMSSPI V9.0-27: VSI OpenVMS SPI for VSI OpenVMS

    © VMS Software Inc., 2025. All rights reserved.

    VMS Software Inc.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI AXPVMS VMSSPI V9.0-27               DISK$FUNYET_SYS:[VMS$COMMON.]

Portion done: 0%...10%...20%...30%...50%...60%...80%...90%...100%

The following product has been installed:
    VSI AXPVMS VMSSPI V9.0-27               Layered Product

VSI AXPVMS VMSSPI V9.0-27: VSI OpenVMS SPI for VSI OpenVMS

    Post-installation tasks are required.

    To start VMSSPI at system boot time, add the following lines to
    SYS$MANAGER:SYSTARTUP_VMS.COM:

        $ file := SYS$STARTUP:VMSSPI$STARTUP.COM
        $ if f$search("''file'") .nes. "" then @'file'
```

```
To shutdown VMSSPI at system shutdown, add the following lines
to SYS$MANAGER:SYSHUTDWN.COM:

    $ file := SYS$STARTUP:VMSSPI$SHUTDOWN.COM
    $ if f$search("'file'") .nes. "" then @'file'
```

5. Post-Installation Tasks

After the installation has successfully completed, the following steps should be performed to configure and start using VMSSPI:

1. Update system start-up and shutdown procedures

   After the installation has successfully completed, include the commands displayed at the end of the installation procedure into SYSTARTUP_VMS.COM and SYSHUTDWN.COM as described below in order to ensure that VMSSPI components are correctly started and stopped when OpenVMS is booted and shutdown.

   a. Modify the site-specific start-up command file, SYS$MANAGER:SYSTARTUP_VMS.COM. After starting all layered products, add the following commands to the end of this file:

      ```
      $ file := SYS$STARTUP:VMSSPI$STARTUP.COM
      $ if f$search("'file'") .nes. "" then @'file'
      ```

   b. Before stopping any layered product, add the following commands to the beginning of site-specific shutdown procedure SYS$MANAGER:SYSHUTDWN.COM:

      ```
      $ file := SYS$STARTUP:VMSSPI$SHUTDOWN.COM
      $ if f$search("'file'") .nes. "" then @'file'
      ```

2. Define VMSSPI logical names

   Before performing the configuration steps described below, run the following command to define logical names used by VMSSPI to determine the location of configuration and messages files:

   ```
   $ @SYS$MANAGER:VMSSPI$LOGICALS.COM
   ```

   This command procedure is run by the VMSSPI start-up procedure; however it is not possible to start VMSSPI until the following configuration tasks have been completed.

3. Create the messages file

   If you do not have an existing installation of VMSSPI then it will be necessary to create an initial messages file before VMSSPI can be started. This is done as follows (assuming that the software is installed onto an ODS-5 formatted file system):

   ```
   $ SET DEFAULT VMSSPI$DATA
   $ COPY MESSAGES^.TXT.TEMPLATE MESSAGES.TXT
   ```

   It you have an existing MESSAGES.TXT file, this will not be overwritten by the installation process, and you will generally not need to perform this step.

   It should be noted that the initial MESSAGES.TXT file will be sufficient to allow you to start VMSSPI to generate an initial configuration; however it will be necessary to modify MESSAGES.TXT in order for VMSSPI to do anything useful, such as sending alerts via email, performing actions in the event of a particular problem being detected, and so on. Details regarding

the format and customization of the MESSAGES.TXT file are provided in *Section 9, "The Messages File"*.

4. Create an initial configuration

   You can now start VMSSPI to generate an initial configuration, or you can run the configuration utility as follows (recommended):

   ```
   $ RUN VMSSPI$ROOT:[BIN]VMSSPI$CONFIGURE_SYSTEM.EXE
   ```

   The configuration utility scans your system and cluster members and generates an initial configuration file. You can now modify this initial configuration file as described in *Section 7, "VMSSPI Configuration File"*.

5. Load license key

   Starting from version 9.0-15, VMSSPI requires a license key from VSI in order to use the software. On an initial installation of VMSSPI, a 90-day node-specific temporary license key will be generated, which may be used for evaluation purposes; however in order to continue using the software after this time you must obtain from VSI either a node-specific license key, or an enterprise license key that can be used on multiple OpenVMS nodes. VMSSPI is licensed on a subscription basis, meaning that license keys have a termination date, after which the subscription must be renewed (and a new key provided by VSI) in order to continue using the software, and to continue receiving support.

   The license mechanism used by VMSSPI is straightforward. License keys are provided as hexadecimal strings that contain encoded details pertaining to the type of license (per node, enterprise, or temporary), node name (if applicable), IP address (if applicable), and expiry date. Keys are stored in the file VMSSPI$DATA:LIC.TXT, which is a simple text file that can be edited using any standard OpenVMS editor to add new license keys, delete expired keys, and so on. Lines in LIC.TXT starting with # or ! are treated as comment lines, and blank lines are ignored. On start-up VMSSPI processes read LIC.TXT and will use the first valid license. If no valid license is found, VMSSPI processes will log an error and will fail to start. This mechanism makes it possible to use a single license file in a cluster, assuming that the file includes valid keys for all cluster nodes (which may be a single enterprise license key or individual per-node keys). Comments may be included to identify which key is for which node, to note expiry dates, and so on.

   If you have received a license key (or keys) with the VMSSPI kit and do not already have an existing VMSSPI installation, after installing VMSSPI the simplest course of action will typically be to start VMSSPI (see the next step) to create an initial LIC.TXT file containing a temporary license key and to then edit this file to include the provided license key. If you are upgrading VMSSPI from a previous version, after installation of the new version it will be necessary to manually create LIC.TXT and to add your license key to it before restarting VMSSPI.

6. Start VMSSPI

   Once you are happy with the configuration and messages files, you can start VMSSPI by manually running the start-up procedure:

   ```
   $ @SYS$STARTUP:VMSSPI$STARTUP.COM
   ```

   Assuming that the start-up was successful, you should see the following three processes running on the system:

   ```
   $ PIPE SHOW SYSTEM | SEARCH SYS$INPUT VMSSPI
   0002E156 VMSSPI$SYSTEM   HIB      4   193846   0 00:02:01.41     16543  13868
   ```

```
   0002E157 VMSSPI$PERFORM  HIB      6    18667   0 00:04:37.19      1266   2109
   0002E158 VMSSPI$SECURITY LEF     14     1271   0 00:00:00.40       989   1402
```

6. VMSSPI Logical Names

You might find the following logical names useful when you implement the VMSSPI in an OpenVMS cluster with multiple system disks.

You can define these logical names in the file SYS$STARTUP:VMSSPI$LOGICALS.COM.

Note that the installation procedure creates an initial version of this file if it does not already exist; however this initial version will likely need to be modified as described below if VMSSPI is used in a cluster context.

6.1. VMSSPI$DATA

The logical name VMSSPI$DATA is used by VMSSPI to specify the location of the configuration and messages files. For OpenVMS clusters with multiple system disks, place the configuration file in a directory that all cluster members can access.

Follow these steps to create define the VMSSPI$DATA logical name for this scenario:

1. Create a directory on a disk that is common to all cluster members, and define the logical VMSSPI$DATA so that it points to this directory. For example:

   ```
   $ DEFINE/SYSTEM/EXECUTIVE_MODE VMSSPI$DATA DISK$COMMON:[VMSSPI]
   ```

   In this example, the logical name `VMSSPI$DATA` points to the `[VMSSPI]` directory on a disk mounted clusterwide with the `DISK$COMMON` logical volume name.

2. Create the file VMSSPI$CONFIGURATION.DAT in this directory. You can do this by using the utility VMSSPI$CONFIGURE_SYSTEM or by copying a previously created configuration file to the directory pointed to by VMSSPI$DATA.

6.2. VMSSPI$NO_CLUSTER_CHECKS

VMSSPI usually runs on every member in an OpenVMS cluster. Only one of the instances performs clusterwide monitoring, which includes clusterwide processes, clusterwide mounted disks and shadow sets, and anything related to queues. The cluster member where the VMSSPI performs this clusterwide monitoring can be any node in the cluster, and is determined by VMSSPI process in the cluster that exclusively holds the VMSSPI$CLUSTER_LOCK lock.

In some situations, you might not want to perform clusterwide monitoring on one particular cluster member, which is the case with the quorum system of a multisite disaster-tolerant cluster. This quorum system usually does not mount any cross-data centre shadow sets and does not have access to the queue file. In a situation like this, disable the clusterwide monitoring on this node by entering the following command prior to starting VMSSPI:

```
$ DEFINE/SYSTEM VMSSPI$NO_CLUSTER_CHECKS TRUE
```

6.3. VMSSPI$IGNORE

This logical name can be defined at SYSTEM level to instruct VMSSPI to ignore one or more specifically monitored processes (as specified in VMSSPI$CONFIGURATION.DAT using the **`PROCESS`** command), as illustrated in the example below, where *`PROC_NAME_1`* and **"***`PROC NAME 2`***"** are the process names of the processes for which monitoring is to be disabled:

```
$ DEFINE/SYSTEM VMSSPI$IGNORE PROC_NAME_1,"PROC NAME 2"
```

Important

If a process names contains spaces, enclose it in double quotation marks.

This facility is intended to provide a mechanism for administrators to temporarily disable monitoring of one or more processes while non-scheduled maintenance operations are performed.

It must be emphasised that this facility applies only to processes explicitly configured for monitoring in the configuration file and not to all processes that might be running on the system.

It should be noted that while monitoring is disabled for processes in this way, VMSSPI will write log entries for the processes in question stating that monitoring is currently disabled. Such log entries will be written for each monitoring cycle until VMSSPI$IGNORE is de-assigned or the process name in question has been removed from the list of process names to be ignored.

6.4. VMSSPI$RERAISE_INTERVAL

This logical name can be used to define a default value for the re-raise interval associated with certain specific alerts. For these alerts, if a re-raise interval is defined and a fault condition persists for longer than is time, the alert will be re-raised, and this behaviour will continue until such time as the fault is corrected. The value specified for VMSSPI$RERAISE_INTERVAL must be an integer value between 0 and 360, where the value represents the number of minutes that a relevant fault condition can remain un-cleared before VMSSPI will re-raise the alert. A value of 0 means that alerts will not be re-raised (unless a specific re-raise interval is otherwise specified in the VMSSPI configuration file for the entity being monitored), and if the specified value is less than or equal to the VMSSPI scan interval, the value will be adjusted to be twice the VMSSPI scan interval (as logically it makes no sense for the re-raise interval to be less than the VMSSPI scan frequency). The VMSSPI$RERAISE_INTERVAL logical name should be defined at SYSTEM level.

The specific alerts that can currently be re-raised in this manner are:

- VMSSPI_ProcessMissing
- VMSSPI_BatchQueueNotStarted
- VMSSPI_BatchJobMissing

Note that it is possible to override the default re-raise interval defined using this logical name or to specify a specific interval using the **/RERAISE** qualifier with the **PROCESS**, **JOB**, and **BATCHQUEUE** commands. The integer value specified with this qualifier is the number of minutes for the interval and the supplied value must conform to the same rules as those described above for the VMSSPI$RERAISE_INTERVAL logical name.

6.5. TCPIP$SMTP_FROM

The logical name TCPIP$SMTP_FROM (as used by VSI TCP/IP Services) can be used to specify an outbound alias that is applied to email messages sent by the VMSSPI SMTP client (all email messages sent be VMSSPI will appear to originate from this address). This facility can be useful in situations where there are constraints imposed by email gateways around where emails can originate and/or from whom. Be aware that this logical name may also be defined for other purposes and care should be taken to ensure that any system-level definition for VMSSPI does not conflict with an existing definition that is relied upon by users or other applications running on the system.

7. VMSSPI Configuration File

VMSSPI needs a configuration file that contains the definition of the OpenVMS items to monitor. In an OpenVMS cluster, the configuration file must be common to all nodes. The name of the configuration file is VMSSPI$CONFIGURATION.DAT and VMSSPI expects to find this file in the directory pointed to by the VMSSPI$DATA logical name. If this logical name is not defined, VMSSPI will fail to start correctly and errors will be reported in the VMSSPI log files.

7.1. Generating the Configuration File

You can easily generate an initial version of the VMSSPI configuration file by running the utility VMSSPI$ROOT:[BIN]VMSSPI$CONFIGURE_SYSTEM.EXE. When you run this utility, a configuration file is generated based on the currently present processes, mounted disks and shadow sets, batch and print queues, and batch jobs.

The generated configuration file allows out-of-the-box monitoring of the OpenVMS environment. To obtain optimal results, however, the system manager will generally need to edit this file.

It is also possible to define a foreign command for VMSSPI$CONFIGURE_SYSTEM.EXE and to run this command with various qualifiers as described below to generate configuration details for specific sub-systems and to write these configuration details to specified files that can then be used to update VMSSPI$DATA:VMSSPI$CONFIGURATION.DAT as necessary with new or changed information. For example, the following commands define a foreign command for VMSSPI$CONFIGURE_SYSTEM.EXE and generate configuration data for shadow sets only, with the resultant output being written to the file SHADOW.DAT:

```
$ VMSSPICFG :== $VMSSPI$ROOT:[BIN]VMSSPI$CONFIGURE_SYSTEM.EXE
$ VMSSPICFG/SHADOW_SETS/OUTPUT=SHADOW.DAT
```

Using your preferred editor, the contents of SHADOW.DAT can then be used as required to update information in VMSSPI$DATA:VMSSPI$CONFIGURATION.DAT.

Permitted qualifiers are summarised in the following table. Note that the **/BATCH_QUEUES** and **/BATCH_JOBS** qualifiers must be specified in full (they *cannot* be abbreviated).

Qualifier	Description
/FULL	Generates a full configuration as would be generated by running VMSSPI$CONFIGURE_SYSTEM.EXE with no options. Note that this qualifier cannot be used in conjunction with any of the other qualifiers listed below, except for **/OUTPUT**. Generating a full configuration is the default.
/OUTPUT=*file-name*	Can be used to specify the name of the output file that is to be created by VMSSPI$CONFIGURE_SYSTEM.EXE. Note that if **/FULL** is used and no output file is specified then output will by default be written to VMSSPI$DATA:VMSSPI$CONFIGURATION.DAT. If no output file is specified and **/FULL** is not specified, output will be written to a file named VMSSPI.TMP in the current working directory.
/PROCESSES	Generates template process monitoring data based on the processes currently running on the system.
/DISKS	Generates template disk monitoring configuration details, including initial hot-file monitoring details.

Qualifier	Description
/SHADOW_SETS	Generates template shadow-set monitoring details (if applicable) for the system in question.
/BATCH_JOBS	Generates template batch job monitoring configuration details based on any currently running or scheduled jobs.
/BATCH_QUEUES	Generates batch queue monitoring configuration details for any batch queues that are defined for the system or cluster in question.
/PRINT_QUEUES	Generates printer queue monitoring configuration details for any print queues that are defined for the system or cluster in question.
/LAN_DEVICES	Generates template configuration details for monitoring of any LAN devices used by the system.
/VERSION	Causes the configuration utility to display the VMSSPI version and exit (no configuration details will be generated).

7.2. Reconfiguring

If changes are made to the VMSSPI$CONFIGURATION.DAT file, the VMSSPI SYSTEM module automatically registers these changes. New items that are added to this file are monitored, and removed items are no longer monitored. However, any such changes will not take effect until the next ALWAYS/NEVER window (midnight by default). To force modifications to the configuration file to take effect immediately, it is necessary to restart VMSSPI.

The PERFORMANCE and SECURITY modules must be restarted for the configuration file changes to take effect.

Important

Only changes to VMSSPI$CONFIGURATION.DAT will be reloaded automatically (and only by the SYSTEM module); any changes made to the MESSAGES.TXT file will not take effect until VMSSPI is restarted. In general, it is recommended that all VMSSPI processes be restarted whenever any changes are made to the configuration in order to ensure that the configuration changes are correctly and fully loaded by all VMSSPI processes.

7.3. Editing the Configuration File

In the configuration file, you need to add entries for the following:

- Cluster name information
- Time interval for performing checks
- Restricted periods for monitoring
- Enabling intrusion detection
- Processes to be monitored
- Disks to be monitored
- Batch queues to be monitored
- Print queues to be monitored

- Batch jobs to be monitored
- Shadow sets to be monitored
- LAN devices to monitor
- Process quota thresholds
- Security filter settings

These configuration file entries are explained in the following sections.

7.3.1. Entering the Cluster Name Information

If are using VMSSPI to monitor an OpenVMS cluster (as opposed to monitoring an individual node), you should specify the name of the cluster in the configuration file. This name is usually the cluster alias, but if no such alias is defined, specify any appropriate name for the cluster in the following format:

```
CLUSTERNAME clustername
```

7.3.2. Defining the Time Interval Between Two Consecutive Checks

The VMSSPI SYSTEM module performs all defined checks at regular time intervals. You can modify the time between two consecutive checks.

To define this interval, enter a value after the **INTERVAL** command in the configuration file:

```
INTERVAL value
```

For *value*, enter an integer to specify the number of seconds between two consecutive checks. The default interval is 60 seconds.

7.3.3. Defining Periods to Restrict Monitoring

You can restrict the monitoring of items to certain time periods. The default generated configuration file defines two default periods: ALWAYS and NEVER.

You can define your own periods by adding the following statements to the configuration file. These periods can then be used in other definitions in the configuration file:

```
PERIOD name_of_period -
    /MONDAY=(BEGIN=hour,END=hour) -
    /TUESDAY=(BEGIN=hour,END=hour) -
    /WEDNESDAY=(BEGIN=hour,END=hour) -
    /THURSDAY=(BEGIN=hour,END=hour) -
    /FRIDAY=(BEGIN=hour,END=hour) -
    /SATURDAY=(BEGIN=hour,END=hour) -
    /SUNDAY=(BEGIN=hour,END=hour) -
    /WORKDAY=(BEGIN=hour,END=hour) -
    /EVERYDAY=(BEGIN=hour,END=hour) -
    /WEEKEND=(BEGIN=hour,END=hour)
```

7.3.3.1. Example

```
PERIOD workhours -
    /WORKDAY=(BEGIN=08:30,END=12:00) -
    /WORKDAY=(BEGIN=13:00,END=17:15) -
    /SATURDAY=(BEGIN=09:00,END=12:30)
```

This example uses the **/WORKDAY** qualifier twice to specify two different periods during a regular work day.

7.3.4. Defining a Process to Monitor

In the configuration file, you can define all the processes to monitor. For each process, specify the **PROCESS** verb followed by the OpenVMS process name and one or more optional qualifiers:

```
PROCESS "process_name" -
[/UIC="uic"] -
[/PERIOD=period] -
[/OCCURRENCES=n] -
[/CLUSTER_WIDE] -
[/NODES=(...)] -
[/ACTION="action"] -
[/RERAISE=n]
```

The *process_name* is the OpenVMS process name, which can have a maximum of 15 characters. It can contain the asterisk (*) and percent (%) wildcards. Also note that process names are case-sensitive. If the process name contains lowercase letters, be sure to place double quotes (" ") around it.

The following table summarises the qualifiers you can use with the **PROCESS** verb.

Qualifier	Description
/UIC	Can be specified in different formats, such as SYSTEM, [SYSTEM], or [1,4]. Specifying the UIC is not mandatory. However, if you have two processes with the same name that run with UICs in different groups, specifying the UIC distinguishes one from the other.
/PERIOD	One of the periods defined earlier in the file. The default value is ALWAYS.
/OCCURRENCES	If the process name contains wildcards, you might want to specify the number of occurrences required for this process, if you have more than one process on your system or cluster with the same fixed name portion and a variable name portion. The default is 1.
/CLUSTER_WIDE	Specifying the qualifier **/CLUSTER_WIDE** indicates that the process should be present on at least one of the nodes of the cluster.
/NODES	Specifies the cluster members on which the process should be present.
/ACTION	Specifies a command (or command procedure) to be executed if VMSSPI detects that the process in question has terminated. This allows actions to be specified on a per-process basis, as opposed to specifying a generic action in MESSAGES.TXT. However, it should be noted that specifying a specific action for a process using the **/ACTION** qualifier does not override any action(s) that might be specified in MESSAGES.TXT for the `VMSSPI_MissingProcess` alert. If there is an action defined in MESSAGES.TXT for an appropriate threshold when an alert is raised, that (generic) action will be performed first, followed by any process-specific action specified for the process using the **/ACTION** qualifier. Note that action commands (generally DCL scripts) may need to include appropriate logic to take into consideration any other details specified for the **PROCESS** command. For example, a failed process may need to be restarted under a specific UIC, or different logic may need to be performed by the action script depending upon the period in which the process failed. If the name of

Qualifier	Description
	a process being monitored includes wildcards, any restart logic should be able to determine exactly which process needs to be restarted, and in clustered environments it may be necessary for any restart logic to ensure that a failed process is restarted on a particular node. Given these comments, it is generally recommended to use the **/ACTION** qualifier judiciously, where actions to restart a failed process (or other entity) are relatively simple or where restarting a failed process (or other entity) is absolutely critical to operations.
/RERAISE	Specifies a re-raise interval after which process-missing alerts for the process in question will be re-raised (assuming process monitoring is active for the process in question). The specified value is the number of minutes that the condition can remain un-cleared before the alert will be re-raised, and the alert will then continue to be re-raised at this interval until the condition is cleared. The specified value must be between 0 and 360, where a value of 0 means that alerts will not be re-raised. If the specified value is less than or equal to the VMSSPI scan interval, the value will be adjusted to be twice the scan interval. Note that specifying **/RERAISE** for a process that can be automatically restarted via an action script is somewhat meaningless. In such situations, after raising an alert, VMSSPI will assume that the action script successfully restarted the process, thereby bypassing the re-raise logic.
/METRICS	Specifies that performance metrics (CPU usage, page faults, direct I/O, and buffered I/O) should be made available to Prometheus monitoring for the process in question. Note that if the process is question is monitored only during specific periods (by specifying the **/PERIOD** qualifier described above), metrics will only be made available during the defined period.

The qualifiers **/CLUSTER_WIDE** and **/NODES** are mutually exclusive. If neither of these qualifiers is present, the process should be available on each node in the cluster.

7.3.4.1. Examples

```
PROCESS ERRFMT /UIC="[1,6]" /PERIOD=ALWAYS
PROCESS ORA* /UIC=ORACLE8 /OCCURRENCES=10
PROCESS QUEUE_MANAGER /UIC=SYSTEM /CLUSTER_WIDE
```

These examples define three processes to be monitored by the VMSSPI SYSTEM module.

7.3.5. Defining Disks to Monitor

In the configuration file, you can define all the disks to be monitored. For each disk, specify the **DISK** verb, followed by the disk specification, and one or more of the following optional qualifiers:

```
DISK disk_name [/PERIOD=period] -
[/CRITICAL=threshold] -
[/MAJOR=threshold] -
[/MINOR=threshold] -
[/WARNING=threshold] -
[/DIO] -
[/QUEUE_LENGTH] -
[/HOTFILES] -
[/NODES=(...)]
```

The *disk_name* is either the physical device name or the logical volume name.

The following table describes the qualifiers you can use with the **DISK** verb. Note that you do not need to define all qualifiers.

Qualifier	Description
/PERIOD	Optional qualifier. The default period is ALWAYS. Note that only the SYSTEM module uses this qualifier.
/CRITICAL, **/MAJOR**, **/MINOR**, and **/WARNING**	Define the amount of free space thresholds on the disk. The values are percentages, which can be floating values. Make sure that the critical threshold is smaller than the major threshold, that the major threshold is smaller than the minor threshold, and so on. These qualifiers are used by the SYSTEM module. To monitor free space on a disk, dynamic volume expansion is taken into account; VMSSPI is capable of monitoring volume sets as well.
/DIO	Enables monitoring of the average number of direct I/Os made to that disk per second. This qualifier is used by the PERFORMANCE module.
/QUEUE_LENGTH	Enables monitoring of the average queue length of direct I/Os per second to that disk. This qualifier is used by the PERFORMANCE module.
/HOTFILES, **/NOHOTFILES**	Enables or disables monitoring of hot files on the disk. This qualifier is used by the PERFORMANCE module. Note that the default configuration file is created with hot file monitoring disabled for all disks. If you want hot file monitoring on one or more disks, you must change the **/NOHOTFILES** qualifier to **/HOTFILES** and restart the PERFORMANCE module.
/NODES	If the disk is not mounted on all cluster members, use **/NODES** to specify the list of nodes that mount the disk. If you do not specify the **/NODES** qualifier, the disk is assumed to be mounted on all cluster members.

When you enable HOTFILE monitoring on one or more disks, you might also want to change two additional parameters in the configuration file:

Parameter	Description
/HOTFILE_INTERVAL	Specifies the time interval over which the average DIO rate on each file will be calculated. The default period is 10 minutes. Specify the time interval in the format "*hh*:*mm*:*ss*". To specify a 5-minute interval, enter the following command: `PARAMETER HOTFILE_INTERVAL/VALUE="00:05:00"`
/HOTFILE_TOPDIO	Specifies the number of hottest files on which VMSSPI will report. The default value is 3. To report on the `5` hottest files, enter the following command: `PARAMETER HOTFILE_TOPDIO/VALUE=5`
/METRICS	Specifies that disk space used (expressed as a percentage of disk capacity) will be made available to Prometheus monitoring for the disk in question. Note that if the disk is question is monitored only during specific periods

Parameter	Description
	(by specifying the **/PERIOD** qualifier), data will only be made available to Prometheus during the defined period.

7.3.5.1. Examples

1. This example defines the monitoring of the disk with the logical volume name of `DISK$KITS`, which is usually mounted on the cluster members `MYCULO` and `SWELL`:

   ```
   DISK DISK$KITS /CRITICAL=5 /MAJOR=10 /NODES=(MYCULO,SWELL)
   ```

   The configuration file that was initially generated using the VMSSPI$CONFIGURE_SYSTEM.EXE utility specifies critical, major, minor, and warning thresholds based on the actual amount of free space on the disk. The actual free space and the physical name of the disk will be added as information in a comment at the end of the line.

2. The following examples show how to define the disk utilization monitoring of two disks. For disk `$1$DKA100`, disk utilization, queue length, and hot files are monitored. For disk `$1$DUA53`, only the queue length is monitored (the **/DIO** and **/HOTFILES** qualifiers are negated):

   ```
   DISK $1$DKA100: /DIO /QUEUE_LENGTH /HOTFILES
   DISK $1$DUA53: /NODIO /QUEUE_LENGTH /NOHOTFILES
   ```

7.3.6. Specifying Batch Queues

Specify the names of batch queues to monitor as follows:

```
BATCHQUEUE queue-name -
[/STARTED_PERIOD=period] -
[/STOPPED_PERIOD=period] -
[/[NO]PENDING_THRESHOLD=...] -
[/[NO]RETAINED_THRESHOLD=...] -
[/NODES=(node...)] -
[/ACTION="action"] -
[/RERAISE=n]
```

Explanations of **BATCHQUEUE** qualifiers are provided below.

Qualifier	Description
/STARTED_PERIOD, **/STOPPED_PERIOD**	Defines when a batch queue is to be "started" during certain periods and "stopped" during other periods. You might want to check whether the actual queue state corresponds with a particular state. Care should be taken when specifying these periods, as overlapping periods may result in the generation of erroneous alert messages. In this regard it should be noted that periods (as defined using the **PERIOD** command) have a resolution of 1 second.
/PENDING_THRESHOLD	Makes it possible to warn the system manager about possible contentions in queues and problems with jobs currently executing and blocking the execution of other jobs. VMSSPI checks if the number of pending jobs in a queue exceeds a specified threshold. If it does, a message is sent.
/PENDING_RETAINED	VMSSPI checks if the number of jobs retained in a queue exceeds a specified threshold. If it does, a message is sent.

Qualifier	Description
/NODES	In most situations, you would not specify the **/NODES** qualifier because queues are usually monitored clusterwide.
/RERAISE	Specifies a re-raise interval after which stopped batch queue alerts for the queue in question will be re-raised (assuming that queue monitoring is active). The specified value is the number of minutes that the condition can remain un-cleared before the alert will be re-raised, and the alert will then continue to be re-raised at this interval until the condition is cleared. The specified value must be between 0 and 360, where a value of 0 means that alerts will not be re-raised. If the specified value is less than or equal to the VMSSPI scan interval, the value will be adjusted to be twice the scan interval.
/ACTION	Specifies a command (or command procedure) to be executed if VMSSPI detects that the batch queue in question is stopped. This allows actions to be specified on a per-queue basis, as opposed to specifying a generic action in MESSAGES.TXT for the `VMSSPI_BatchQueueNotStarted` alert. However, it should be noted that specifying an action for a specific batch queue using the **/ACTION** qualifier does not override any action(s) specified in MESSAGES.TXT for the `VMSSPI_BatchQueueNotStarted` alert.

If the two threshold qualifiers are negated, those items are not monitored.

Pending and retained jobs are monitored during the started and stopped periods that are defined.

7.3.6.1. Example

```
BATCHQUEUE SYS$BATCH
```

In this example, the batch queue `SYS$BATCH` is always supposed to be started. If the state of the batch queue differs from idle, busy, or available, a notification is sent. Pending and retained jobs are not monitored.

7.3.7. Specifying Print Queues

Specify the names of print queues whose actual status needs to be monitored as follows:

```
PRINTQUEUE queue-name [/PERIOD=time_period] -
[/[NO]PENDING_THRESHOLD=...] -
[/[NO]RETAINED_THRESHOLD=...] -
[/NODES=(node...)]
```

The *queue-name*, which is required, can contain asterisk (*) wildcards. Explanations of **PRINTQUEUE** qualifiers are provided in the following table:

Qualifier	Description
/PERIOD	The default value is ALWAYS.
/PENDING_THRESHOLD	Warns the system manager about possible contentions in queues and problems with jobs currently executing and blocking the execution of other jobs. When the number of pending jobs in a queue exceeds a specified threshold, a message is sent.

Qualifier	Description
/RETAINED_THRESHOLD	When the number of jobs retained in a queue exceeds a specified threshold, a message is sent.
/NODES	In most situations, you would not specify the **/NODES** qualifier because queues are usually monitored clusterwide.

If the two threshold qualifiers are negated, those items are not monitored.

Pending and retained jobs are monitored during the "started" and "stopped" periods that are defined.

7.3.7.1. Example

```
PRINTQUEUE SYS$LTA* /PERIOD=WORKHOURS
```

This example specifies that the state of any print queue whose name begins with `SYS$LTA` is to be monitored only during the time intervals defined by the period `WORKHOURS`.

7.3.8. Specifying Batch Jobs

Specify the names of batch jobs whose actual status needs to be monitored as follows:

```
JOB job-name [/USERNAME=username] -
[/QUEUE=queuename] -
[/PERIOD=time_period] -
[/NODES=(...)] -
[/CHECK=OVERRUN] -
[/RERAISE]
```

The *job-name* is required. It defines the batch jobs that VMSSPI must check; it can contain the asterisk (*) and percent (%) wildcards.

The following table describes the qualifiers you can use with the JOB verb:

Qualifier	Description
/USERNAME	Optional qualifier. The default is SYSTEM.
/QUEUE	Optional qualifier. The default is SYS$BATCH. Specify the generic or execution queue on which the batch job is entered. If a generic queue is specified, and the job is currently running on one of the execution queues, the job will be considered to be present.
/PERIOD	Optional qualifier. The default is ALWAYS.
/NODES	In most situations, you would not specify this qualifier because batch jobs are usually monitored clusterwide.
/RERAISE	Specifies a re-raise interval after which missing batch job alerts for the job in question will be re-raised (assuming that job monitoring is active). The specified value is the number of minutes that the condition can remain un-cleared before the alert will be re-raised, and the alert will then continue to be re-raised at this interval until the condition is cleared. The specified value must be between 0 and 360, where a value of 0 means that alerts will not be re-raised. If the specified value is less than or equal to the VMSSPI scan interval, the value will be adjusted to be twice the scan interval.
/CHECK=OVERRUN	This option can be specified to instruct VMSSPI to raise an alert if the job in question appears to have overrun the period specified using the

Qualifier	Description
	/PERIOD option. Note that alerts will also be raised if the job is running prior to the start of the defined period, as this could in fact also be an overrun from a previous job submission the previous day. The alert raised by any such overrun condition is `VMSSPI_BatchJobOverrun`.

7.3.8.1. Example

```
JOB DAILY_CLEANUP /USERNAME=SYSTEM /QUEUE=SYS$BATCH
```

This example defines the monitoring of a batch job named `DAILY_CLEANUP` for user SYSTEM, which should be on the `SYS$BATCH` queue at all times.

7.3.9. Specifying Shadow Sets

Specify the names of shadow sets whose actual status needs to be monitored as follows:

```
SHADOWSET shadow-name /MEMBERS=(member1,[member2],[member3]) -
[/PERIOD=period] -
[/NODES=(...)]
```

For *`shadow-name`*, specify the physical name of the shadow set. You must also specify the members. No logical names are accepted.

The following table describes other qualifiers to **SHADOWSET**:

Qualifier	Description
/MEMBERS	Used to specify all the shadow set members.
/PERIOD	Optional qualifier. The default value is ALWAYS.
/NODES	Needs to be specified only if the shadow set is not mounted on all cluster members.

7.3.9.1. Example

```
SHADOWSET DSA1 /MEMBERS=($1$DGA1000,$1$DGA2000)
```

This example tells the VMSSPI to monitor whether the shadow set `DSA1` contains the members `$1$DGA1000:` and `$1$DGA2000:` at all times.

7.3.10. Specifying a Rules File Using INCLUDE

Using a rules file allows you to keep the rules of different monitored objects in separate files so that they can be included or removed as required.

You can specify a file containing rules for objects to be monitored as follows:

```
INCLUDE=rules-file-name
```

For *`rules-file-name`*, specify the name of a file containing rules for objects that are to be monitored. To specify multiple rules files, use a different `INCLUDE` statement for each one. Add one or more `INCLUDE` statements to your main CONFIGURATION.DAT file to enable rules files that you select.

7.3.10.1. Example

```
INCLUDE=SYS$MANAGER:DISK$CONFIGURATION.DAT
```

7.3.11. Enabling Intrusion Detection

You can enable intrusion detection by adding the following statement to the configuration file:

```
INTRUDERS
```

After you make this addition to the configuration file, users who enter an incorrect password four times are considered to be intruders and are denied access to the system even if they then enter a correct password. A message is also sent to signal a suspected intruder.

Optionally, you can also send a notification of `SUSPECTS`. You specify `SUSPECTS` by entering the number of login attempts a user is permitted, which defines the threshold you want to establish on your system. You define this threshold as part of the `INTRUDER` statement in the configuration file, which also specifies that you want to include notification of `SUSPECTS`. The format for this is as following:

```
INTRUDERS/INCLUDE=SUSPECTS/THRESHOLD=n
```

The default threshold is *n*=0.

7.3.11.1. Example

```
INTRUDERS/INCLUDE=SUSPECTS/THRESHOLD=3
```

This example instructs VMSSPI to do the following:

- Send a message to the Message Agent when either an intruder or a suspect is detected.
- Update the message when the count of the intruder or suspect changes.
- Clear the message when the intrusion record is deleted from the intrusion database.

7.3.12. LAN Devices to Monitor

Use the following format to enable monitoring of a LAN device:

```
LAN device-name/NODE=node-name
```

where:

- *device-name* is the name of the Ethernet controller that you want to monitor.
- **Optional:** `/NODE=`*node-name* is the name of the node on which the controller is located. This part is optional and only applicable if you are running in a cluster.

7.3.12.1. Example

```
LAN EWAO: /NODE=SWELL
```

7.3.13. Security Filter Setting

By default, the SECURITY module processes all messages that the AUDIT_SERVER writes to the listener mailbox.

If you need to publish only certain classes of events, you can enable additional filtering. To do this, edit the configuration file, disabling those classes of security events that do not need to be published by removing the comment sign (the exclamation point (`!`)) from selected commands:

```
!DISABLE AUDIT
!DISABLE BREAKIN
```

```
!DISABLE INSTALL
!DISABLE LOGFAIR
!DISABLE LOGIN
!DISABLE MOUNT
!DISABLE NETPROXY
!DISABLE SYSUAF
!DISABLE RIGHTSDB
!DISABLE SYSTIME
!DISABLE SYSGEN
!DISABLE OBJ_CREATE
!DISABLE OBJ_DELETE
!DISABLE OBJ_ACCESS
!DISABLE CONNECTION
!DISABLE NCP
!DISABLE PROCESS
```

8. Log Files

VMSSPI creates a number of log files that are the primary source of information (subject to the level of logging that is enabled) about what the VMSSPI is monitoring, what it has detected, any problems that have been encountered, and what information has been sent to consumers.

The following table lists the log files created by each of the VMSSPI modules, where node-name will be replaced by the name of the OpenVMS node. If unexpected VMSSPI behaviour is observed, these files should be checked for any information that may be relevant.

Module	**Log file**
SYSTEM	VMSSPI$LOG:VMSSPI$SYSTEM_*node-name*.OUT
PERFORMANCE	VMSSPI$LOG:VMSSPI$PERFORMANCE_*node-name*.OUT
SECURITY	VMSSPI$LOG:VMSSPI$SECURITY_*node-name*.OUT

Note that by default the logical name VMSSPI$LOG is defined in the command procedure SYS$STARTUP:VMSSPI$LOGICALS.COM to point to the directory VMSSPI$ROOT:[LOG]. If the logical name VMSSPI$LOG is not defined when VMSSPI is started, the VMSSPI start-up procedure will define it as SYS$MANAGER. If you do not want log files to be written to either of these locations, you may modify the definition in SYS$STARTUP:VMSSPI$LOGICALS.COM as appropriate.

9. The Messages File

The messages file VMSSPI$DATA:MESSAGES.TXT defines the characteristics and properties of all alerts that can be raised by any of the VMSSPI modules, and the file must contain an entry for every such alert message type. Additionally, the messages file describes details about how alerts will be reported, be it via email, or via a service such as DataDog or Dynatrace. The following text describes the format of this file and how it can be customized to meet particular operational requirements.

The messages file can essentially be divided into two sections, with the first section containing definitions for various items such as mailing lists, SMTP gateway details, organizational information, and the specification for the specific alerting module that is to be used. In general, the order in which these items are specified in the messages file is not important; however it is recommended to use the layout used in the template file provided with the VMSSPI kit. It should also be noted that some items (such as the SMTP gateway port) are optional (and defaults will be used); however it is recommended that all items are defined.

Note that all commands and identifiers specified in the messages file are case-sensitive and all commands must be terminated with a semi-colon. Quoted string values may span multiple lines and can contain single quotation marks, but may not contain double quotation marks.

- The first items to be defined in the messages file are the `organization code` and the `organization name`, as illustrated in the following example. An organization code must be specified, and can be any short sequence of characters; specifying the organization name is optional, but desired:

  ```
  set organization code jsl;
  set organization name "John Smith Ltd.";
  ```

  The values specified for organization code and name are supplied to altering modules, and can be sent as part of the alert message. It should be noted that this information is not relevant to any of the current altering modules; however it is possible that in the future VSI will provide a service to monitor customers systems via VMSSPI as an extension of existing VSI support services, whereupon the organization code will be required (along with other details) and will map to the organization code used by the VSI Service Platform (https://sp.vmssoftware.com/).

- After specifying organization details, the next entries in the messages file would typically be the name (or address) and port number used by the SMTP gateway for sending alerts via email, as illustrated in the following example. If the SMTP port is not specified, then the default value of 25 will be used:

  ```
  # Define SMTP gateway
  #
  set smtp host "biggles.vmssoftware.com";
  set smtp port 25;
  ```

  Note that the VMSSPI SMTP mail facility does not currently support SSL/TLS communication with the gateway; however this functionality will be included in a future release.

- After specifying the SMTP gateway details, you can define zero or more mailing lists. There can be any number of mailing lists. Each such list must have a unique name and may specify one or more email addresses and a subject. The list of email addresses must be specified as a comma-separated list enclosed in double quotation marks. The following example illustrates these points, defining a mailing list called `ML1`:

  ```
  define mailing list ML1 "bill.smith@example.com,
  bill.smith@dummy.com"
  subject "There was a problem";
  ```

  The mailing list `ML1` can now be assigned to zero or more alerts, as described below. Mailing list names do not need to use upper-case characters; however using upper-case characters can be useful, as all reserved words within the messages file are lower-case. It should also be noted that the same email addresses may be used for multiple mailing lists if so desired.

- Events detected and reported by VMSSPI can be assigned one of five classifications, namely "none", "operational", "performance", "security", or "availability", with the default being "none" (unclassified). The classification is primarily useful to services such as Splunk, where it can be used to help filter events, however it is also possible to use the **`disable`** command in the messages file as shown below to explicitly disable the reporting of any events with the stated classification:

  ```
  # Disable reporting of security and performance events…
  disable classification performance;
  ```

```
disable classification security;
```

This can be useful if, for example, you are interested only in certain sets of events (as opposed to disabling individual events). Note, however, that all events will still be detected by VMSSPI, and emails will be sent and actions executed (if configured), regardless of this setting. It should also be mentioned that the assignment of classifications in the messages file is left to the administrator, as events that might (for example) be considered relevant to security by one organisation might be considered (for example) more of an operational matter to another organisation.

- The next item to be defined in the messages file is the alerting module(s) that you wish to use to publish alert data. It is possible to specify a maximum of two alerting modules, and by default, the `null` module is defined, which does nothing:

  ```
  use interface module "vmsspi$root:[lib]vmsspi$null_shr.exe" "";
  ```

  In order to use the Slack (vmsspi$slack_shr.exe) or DataDog (vmsspi$datadog_shr.exe) modules, it is necessary to specify an API key as part of the service endpoint URL[1], while for Dynatrace (vmsspi$dynatrace_shr.exe) it is necessary to specify the service endpoint and API key as separate parameters, as illustrated (but commented out) in the messages template file included with the VMSSPI kit. For Dynatrace it is also possible to specify a third argument, which is the endpoint for log ingestion; if this parameter is supplied then the Dynatrace module will publish details of all events to the Dynatrace log in addition to raising an alert. The "file" module (vmsspi$file_shr.exe) simply logs alert messages to VMSSPI log files, and the MQTT module (vmsspi$mqtt_shr.exe) can be used to publish alerts to an MQTT broker, whereupon they can be consumed by interested applications. A more detailed description of some of these modules is provided elsewhere in this document.

  Note that if you wish only to receive alerts via email, it is still necessary to define the "null" module. It should also be noted that the sending of alert details via email is independent of the alerting module: if a mailing list is specified for a given alert type, emails will be always be sent for that alert type.

- The remaining entries in the messages file are the definitions for all possible messages/alerts that can be reported by the VMSSPI product. You are free to customize most message details; however there are some restrictions that will be discussed below, and in particular it is most important to note that no message definitions should be deleted (or commented out), and message names should not be changed. If messages are deleted or their names are changed, VMSSPI will not be able to raise alerts for any such messages. The syntax of these message definitions is for the most part straightforward; however there are some subtle points that need to be considered, and there are various default and enumerated values that need to be explained.

  The following message definition illustrates most of the points that need to be considered (note that the `text` line has wrapped):

  ```
  VMSSPI_ComputableProcesses = {
          subject         = "Number of computable processes is above threshold.";
          severity        = unknown;
          facility        = VMSSPI;
          alias           = VMSSPI_ComputableProcesses;
          mailing list    = none;
          throttle        = 1;
          action          = "";
          text            = "Average number of processes in COM or COMO state is
                             above <$THRESHOLD>.";
          classification  = performance;
  ```

[1] Note that for DataDog, instead of specifying an endpoint URL it is possible to instead specify DataDog email address to receive alert messages.

```
        disable         = false;
        thresholds      = {
                [
                         value > 8.000000,
                         critical,
                         action = ""
                ],
                [
                         value > 6.000000,
                         major,
                         action = ""
                ],
                [
                         value > 4.000000,
                         minor,
                         action = ""
                ],
                [
                         value > 2.000000,
                         warning,
                         action = "",
                         text = "Alternative per-threshold message text"
                ]
        }
        notes           = "";
}
```

The first point to note is that every message has a name (`VMSSPI_ComputableProcesses` in the above example), and by convention these names beginning with `VMSSPI_`. As commented previously, it is important not to change these names, as in the event that VMSSPI needs to issue an alert that is associated with the message name in question, the name will not be found and consequently the alert will not be raised (and VMSSPI will log an error). The following table describes the attributes that can be specified for each message. Note that while it is possible to not specify some message attributes, it is generally recommended to include all attributes in the definition and to use default values for those attributes for any attributes that may not be relevant or important.

Attribute	Description
`subject`	This field provides a subject description for the message. Note that this is not intended to be a description of the particular situation that was detected, but rather a short sentence of text that would be suitable for an email subject line. If a mailing list is specified for the message and the subject text is not an empty string ("") then this text will be used for the subject when sending email messages. If the subject text is an empty string then the mailing list subject text will be used when sending emails. In general it is recommended to specify a value for this field. As of VSI OpenVMS VMSSPI V9.0-6, it is possible to specify the keyword `text` for the subject description. If this is done for a particular message then the message text (determined by the `text` field) will be used for the subject. This facility can useful when publishing messages to services such as Dynatrace and PagerDuty, or for if more meaningful subject lines are desired for email messages sent by VMSSPI.
`severity`	This field specifies the *default* severity of the situation associated with the message. Permitted values for this field are `unknown`, `minor`, `warning`, `normal`, `major`, and `critical`. There is clearly some ambiguity between some of the values (such as `major` and `critical`);

Attribute	Description
	however, in such situations the difference between one and another of these values is generally of no particular consequence, as they essentially have comparable meaning. The reasons for this set of severity codes are largely historical. As noted, this field specifies the *default* severity for the event in question, and this can be overridden by specifying different severity values for different alerting thresholds, as shown in the example above. If no thresholds are defined or if a severity is not specified for a particular threshold then the default severity will be used.
`facility`	This field currently serves no real purpose and should always be assigned the value `VMSSPI`.
`alias`	This field also serves no real purpose in the current implementation of VMSSPI for VSI OpenVMS, and the assigned value should simply match the name of the message. It is anticipated that future versions of VMSSPI will make use of this field.
`mailing list`	This field can be used to specify a mailing list for the message. The mailing list must have been previously defined as described above. In the event of this message/alert being issued by VMSSPI, an email will be sent to each email address specified in the mailing list. If you do not wish to specify a mailing list then the reserved word none should be used, as shown in the above example.
`throttle`	This field can be used to control the frequency with which the message/alert in question will be reported by VMSSPI. This can be useful for preventing message storms, where a particular event occurs that might otherwise cause VMSSPI to send potentially large numbers of alert messages to the monitoring infrastructure or via email within a short period of time. The value for this field is an integer that defines the number of times the message/alert in question will be reported per minute. For the example above, the message/alert would be reported only once within a 60s period, irrespective of how many times the event actually occurred within this time window. A value of 0 can be specified to disable throttling. Note that for some messages/alerts (for example those pertaining to the monitoring of entities for which there are multiple instances, such as disks), it may be advisable to set the throttle to 0 or to a value at least equal to the number of such entities in question. For example, with throttle set to 1 for `VMSSPI_FreeSpace`, if there were issues with multiple disks at roughly the same time, only one alert (for one disk) may be raised, which would not provide a complete and accurate picture of the problem scope.
`action`	This field can be used to specify a default action to be taken when the associated event is detected by VMSSPI. The action is specified as a quoted string and may be any valid OpenVMS command. Most commonly, the action would be to invoke a DCL command procedure. For example, if VMSSPI detected that a monitored process had failed, an action script could be used to restart the process. It should be noted that any variable values used by VMSSPI when formatting the message

Attribute	Description
	text string will also be passed as parameters to action scripts, whereupon such values may be used as necessary. For example, for the above example the threshold value determined by VMSSPI would be supplied as input to an action script. In the case of a failed process, the name of the process would be supplied to the action script, thereby allowing the script to determine which process potentially needs to be restarted. As discussed previously (and below) it is also possible to specify individual action scripts for any thresholds that are defined, and these will be used in preference to the default action script, if defined. If no action script is defined, the value for the `action` field should be set to an empty quoted string.
`text`	This field defines that text that will be logged for the message/alert in question, and that will be sent via email if a mailing list is defined. The message text should be brief and to the point, typically comprising a single sentence. It should be noted that some message text strings specify values that will be populated at runtime by VMSSPI to provide specific details about what has occurred or the entity to which the event related, such as a process name, the name of the user that triggered a security event, and so on. In some situations, the entire message text will in fact be generated by VMSSPI on the fly. These variable elements have the general form `<$VARIABLE>`, where `VARIABLE` is the name of a variable part of the string that VMSSPI will fill in dynamically. It is important not to remove any of these variable definitions or to change the variable names, as this will cause VMSSPI to incorrectly report events; however, the surrounding text (if any) may be changed. While you can add additional variables to the message text, doing so is pointless unless VMSSPI is aware of the variables in question and records the associated values.
`classification`	This field can be used to specify a general category for each event type, with permitted values being `none`, `performance`, `security`, `operational`, and `availability`. There are no specific rules about how these values should be used, and you should set values for each message that make the most sense for your environment. These values are sent by interface modules to the monitoring infrastructure and may be used to help with searching and grouping of events. The classification will also be included in emails. Note that it is also possible to disable the reporting of events by their classification, as described elsewhere in this document.
`disable`	This field can be used to disable the reporting of the specific event type in question (in contrast to disabling entire classifications). To disable a particular event type, set the value to `true` (the default value is `false`, as used in the example above this table).
`thresholds`	This field can be used to specify a comma-separated list of up to 8 thresholds for the alert/message in question, where each threshold definition is enclosed inside square brackets and consists of a value, a severity, an action, and an optional alternative per-threshold message text string, as illustrated in the above example. Threshold levels be expressed as floating-point values, and these can be compared by VMSSPI with observed values using the ">", "<", and "=" operators. VMSSPI will first check for thresholds specifying exact matches ("="), followed by ">",

Attribute	Description
	and "<". If a message has no thresholds, the values of the thresholds field should be set to {}. If an alternative message is specified for a threshold, this will be used in place of the value specified for the `text` attribute, and may contain the same or different variable attributes (assuming such attributes are available for the message in question). Per-threshold messages are most useful in binary situations, where (for example) a threshold value of 1.0 indicates failure and a threshold value of 0.0 indicates that normal service has been resumed.
`notes`	This field can be used to specify notes associated with the message in question, and any such text will be included in the body of emails sent by VMSSPI in addition to other message details. Such notes might for example describe actions to be taken when the event in question occurs, and so on. The notes text must be enclosed in double quotes and can span multiple lines. The text should not include double quotes, but may use single quotes.

9.1. Defined Messages

9.1.1. VMSSPI_AUDSRV_Audit

Type:	Message
Description:	One of the characteristics of the security auditing system has been changed, or a change in the list of auditable events has been made.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Minor

9.1.2. VMSSPI_AUDSRV_AuditServer

Type:	Message
Description:	The AUDIT_SERVER process is not running, or security auditing is currently disabled on this system.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Critical

9.1.3. VMSSPI_AUDSRV_Breakin

Type:	Message
Description:	A break-in attempt was detected.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Critical

9.1.4. VMSSPI_AUDSRV_Connection

Type:	Message
Description:	A logical link was established or terminated through DECnet, DECwindows, IPC, or SYSMAN.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.5. VMSSPI_AUDSRV_Install

Type:	Message
Description:	A change was made to the known file list through the INSTALL utility.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.6. VMSSPI_AUDSRV_Logfailure

Type:	Message
Description:	A Login failure was detected.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None

Severity:	Major

9.1.7. VMSSPI_AUDSRV_Login

Type:	Message
Description:	A successful process login was detected.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Normal

9.1.8. VMSSPI_AUDSRV_Logout

Type:	Message
Description:	A process logout was detected.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Normal

9.1.9. VMSSPI_AUDSRV_Mount

Type:	Message
Description:	A device was mounted or dismounted.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.10. VMSSPI_AUDSRV_NCP

Type:	Message
Description:	Access to the network configuration database through the network control program (NCP) was made.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis

SPI Config:	None
Severity:	Warning

9.1.11. VMSSPI_AUDSRV_Netproxy

Type:	Message
Description:	The network proxy authorization file has been modified.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.12. VMSSPI_AUDSRV_ObjectAccess

Type:	Message
Description:	An object (for example, a file, device, volume or queue) has been accessed.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.13. VMSSPI_AUDSRV_ObjectCreate

Type:	Message
Description:	An object (for example, a file, device, volume, or queue) has been created.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.14. VMSSPI_AUDSRV_ObjectDelete

Type:	Message
Description:	A device has been deleted.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis

SPI Config:	None
Severity:	Minor

9.1.15. VMSSPI_AUDSRV_Process

Type:	Message
Description:	A process control system service has been executed.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.16. VMSSPI_AUDSRV_Rightslist

Type:	Message
Description:	The rights list database has been modified.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.17. VMSSPI_AUDSRV_Sysgen

Type:	Message
Description:	One or more SYSGEN parameters have been changed.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.18. VMSSPI_AUDSRV_Systime

Type:	Message
Description:	The system time has been changed.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis

SPI Config:	None
Severity:	Minor

9.1.19. VMSSPI_AUDSRV_Sysuaf

Type:	Message
Description:	The system user authorization file has been changed.
Frequency:	As occurs
Used By:	Security module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.20. VMSSPI_ActiveCPU

Type:	Message
Description:	Not all available CPUs are currently active.
Frequency:	Every 2 minutes
Used by:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.21. VMSSPI_BatchQueue

Type:	Message
Description:	A batch queue problem has been detected. The configuration file defines a batch queue to monitor, but no information could be obtained on this queue (for example, the queue does not exist, or the queue manager is not running).
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Minor

9.1.22. VMSSPI_ClusterMemberAdded

Type:	Message
Description:	A new member has been added to the OpenVMS cluster.
Frequency:	As defined by the INTERVAL parameter

Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Normal

9.1.23. VMSSPI_ClusterMemberRemoved

Type:	Message
Description:	A member has been removed from the OpenVMS cluster.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Critical

9.1.24. VMSSPI_DiskState

Type:	Message
Description:	A problem has been detected with one of the disks to be monitored. Possible messages are: • Device is not a disk (warning) • Disk is not mounted (critical) • Disk is mounted foreign (warning) • Disk is allocated (warning) • Disk is marked for dismount (critical) • Disk is spare for a software RAID set (warning) • Disk is a member of a shadow set (warning) • Disk is a member of a software RAID set (warning) • Mount verification is in progress (critical) • Mount verification has timed out (critical) • Mount verification is pending (critical)
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On node basis
SPI Config:	None

Severity:	Depends on actual message sent; see above

9.1.25. VMSSPI_FreeSpace

Type:	Message
Description:	The free space on a disk can be monitored by specifying the disk in the configuration file, together with 4 thresholds. Those thresholds correspond with the critical, major, minor and warning severity levels. Each time the actual free space on the disk drops below a new threshold, a message is sent.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Corresponding to the defined thresholds.

9.1.26. VMSSPI_HardwareError

Type:	Message
Description:	The error count on a device, CPU, or Memory has changed. Note that when the SYSTEM module is started, no messages are sent regarding the current error count on devices.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.27. VMSSPI_Intruder

Type:	Message
Description:	An intruder has been detected on the system.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Critical

9.1.28. VMSSPI_LANCarrierCheckFailures

Type:	Message
Description:	Carrier failures have been detected on a LAN device.

Frequency:	Every minute
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.29. VMSSPI_MemberState

Type:	Message
Description:	A problem has been detected with one of the members of a shadow set: • Disk is unexpected member (minor) • Disk is copy-target (minor) • Disk is merge member (warning) • Disk is in mount-verification (critical) • Mount verification has timed out for the disk (critical)
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Depends on actual message sent; see above

9.1.30. VMSSPI_NPAGEDYN

Type:	Message
Description:	Non-paged pool has been successfully expanded (warning), or the system failed to expand non-paged pool.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On node basis
SPI Config:	None
Severity:	Depends on actual message sent; see above

9.1.31. VMSSPI_PendingJobs

Type:	Message
Description:	The number of jobs on a print or batch queue waiting for execution exceeds a specified threshold.
Frequency:	As defined by the INTERVAL parameter

Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Minor

9.1.32. VMSSPI_PrintQueue

Type:	Message
Description:	A problem has been detected with the status of a print queue. Either no information could be obtained on the queue (queue does not exist or the queue manager is not running) or the queue has not been started (the status is not idle or busy).
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Minor

9.1.33. VMSSPI_ProcessLooping

Type:	Message
Description:	A process has been detected doing no direct or buffered I/O, while using at least 25% of one CPU for a period of at least two minutes. A message is then sent to indicate that the process seems to be looping.
Frequency:	Every 2 minutes
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.34. VMSSPI_ProcessOccurrences

Type:	Message
Description:	The configuration file defines a process that is to be monitored; the process name contains wildcards; and a certain number of occurrences of this process are present. The OpenVMS SPI detects a number of those processes but not a sufficient number of the occurrences that were defined.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On node basis or clusterwide
SPI Config:	None
Severity:	Major

9.1.35. VMSSPI_ProcessState

Type:	Message
Description:	A process has been detected in a state other than LEF, LEFO, CEF, HIB, HIBO, COM, and CUR.
Frequency:	Every 10 seconds. The process must be seen in that special state during a period of minimum one minute.
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.36. VMSSPI_QueueManager

Type:	Message
Description:	The status of a queue manager is not running.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Critical

9.1.37. VMSSPI_Quota_ASTLM

Type:	Message
Description:	A process has used most of its ASTLM quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.38. VMSSPI_Quota_BIOLM

Type:	Message
Description:	A process has used most of its BIOLM quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None

Severity:	Major

9.1.39. VMSSPI_Quota_BYTLM

Type:	Message
Description:	A process has used most of its BYTLM quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.40. VMSSPI_Quota_DIOLM

Type:	Message
Description:	A process has used most of its DIOLM quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.41. VMSSPI_Quota_ENQLM

Type:	Message
Description:	A process has used most of its ENQLM quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.42. VMSSPI_Quota_FILLM

Type:	Message
Description:	A process has used most of its FILLM quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None

Severity:	Major

9.1.43. VMSSPI_Quota_PGFLQUOTA

Type:	Message
Description:	A process has used most of its PGFLQUOTA quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.44. VMSSPI_Quota_PRCLM

Type:	Message
Description:	A process has used most of its PRCLM quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.45. VMSSPI_Quota_TQELM

Type:	Message
Description:	A process has used most of its TQELM quota.
Frequency:	Every 10 seconds
Used By:	Performance module
Scope:	On node basis
SPI Config:	None
Severity:	Major

9.1.46. VMSSPI_RetainedJobs

Type:	Message
Description:	The number of retained jobs on a print or batch queue exceeds a given threshold.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None

Severity:	Major

9.1.47. VMSSPI_SecurityServer

Type:	Message
Description:	The Security Server is not active. Intrusion detection is currently disabled.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On a per-node basis
SPI Config:	None
Severity:	Critical

9.1.48. VMSSPI_ShadowSetState

Type:	Message
Description:	A problem has been detected with the state of a shadow set. The shadow set is either not mounted or is in mount verification.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On node basis
SPI Config:	None
Severity:	Critical

9.1.49. VMSSPI_Started

Type:	Message
Description:	One of the OpenVMS SPI processes has been started.
Frequency:	Upon start-up
Used By:	All SPIs
Scope:	On node basis
SPI Config:	None
Severity:	Normal

9.1.50. VMSSPI_BatchJobMissing

Type:	Monitor
Description:	A batch job is missing.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide

SPI Config:	None
Severity:	As defined in the SPI template. The default is critical.

9.1.51. VMSSPI_BatchJobOverrun

Type:	Monitor
Description:	A batch job appears to be overrunning the defined window.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Cluster-wide
SPI Config:	None
Severity:	As defined in the SPI template. The default is warning.

9.1.52. VMSSPI_BatchQueueNotStarted

Type:	Monitor
Description:	The status of a batch queue is other than "available", "idle", or "busy".
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	As defined in the SPI template. The default is major.

9.1.53. VMSSPI_BatchQueueNotStopped

Type:	Monitor
Description:	The status of a batch queue is other than stopped.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	As defined in the SPI template. The default is major.

9.1.54. VMSSPI_CPUutilization

Type:	Monitor
Description:	The system uses a lot of CPU.
Frequency:	1 minute
Used By:	Performance module
Scope:	On node basis

Frequency:	Every minute
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the actual CPU utilization over total CPU capacity times 100, calculated over a period of 10 minutes. The value is thus independent of the number of processors in your system. Verify the additional conditions defined in this template and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: 95% ● Major: 90% ● Minor: 85% ● Warning: 80% In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	As defined in the template

9.1.55. VMSSPI_DiskNotAvailable

Type:	Monitor
Description:	A disk to monitor is defined in the configuration file, but this disk appears to be unavailable.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On node basis
SPI Config:	None
Severity:	Warning

9.1.56. VMSSPI_DiskQueueLength

Type:	Monitor
Description:	High queue length of DIOs detected on one of the disks.
Frequency:	Every minute
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the average queue length per second to the disk, over a period of 10 minutes. Verify the additional conditions defined in this template, and adjust the severity levels according to your preferences. The default severity levels in this template are:

	• Critical: more than 15 outstanding I/Os per second • Major: more than 10 outstanding I/Os per second • Minor: more than 7 outstanding I/Os per second • Warning: more than 2 outstanding I/Os per second In some cases, you might need to define the severity levels per system or per disk type, which requires you to create additional conditions for this template.
Severity:	As defined in the template

9.1.57. VMSSPI_DiskWriteLocked

Type:	Monitor
Description:	A disk is mounted read-only.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On node basis
SPI Config:	None
Severity:	Minor

9.1.58. VMSSPI_Hotfile

Type:	Monitor
Description:	A hot file has been detected. The number of DIOs per second made to this file is above the threshold.
Frequency:	As defined by the HOTFILE_INTERVAL parameter. The default is 10 minutes.
Used By:	System module
Scope:	Clusterwide
SPI Config:	The value reported is the average number of DIOs per second made to this file. Verify the threshold defined in the template, and modify it according to your preference. The default threshold is 500 DIOs per second. In some cases, a different threshold might be required for each system or disk.
Severity:	Critical

9.1.59. VMSSPI_LANutilization

Type:	Monitor
Description:	The throughput on a LAN device is high.
Frequency:	Every minute.
Used By:	Performance module
Scope:	On node basis

SPI Config:	The value reported is the line utilization to the line speed times 100 for the last minute. Verify the additional conditions defined in this template, and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: throughput is above 55% of line capacity. ● Major: throughput is above 50% of line capacity. ● Minor: throughput is above 45% of line capacity. ● Warning: throughput is above 40% of line capacity. In some cases, you might need to define the severity levels per system or per device, which requires you to create additional conditions for this template.
Severity:	As defined in the template

9.1.60. VMSSPI_MemoryUtilization

Type:	Monitor
Description:	High utilization of physical memory.
Frequency:	Every minute.
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the average percentage of memory utilization over a time period of 5 minutes. Verify the additional conditions defined in this template, and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: more than 95% of the physical memory is in use. ● Major: more than 90% of the physical memory is in use. ● Minor: more than 85% of the physical memory is in use. ● Warning: more than 80% of the physical memory is in use. In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	As defined in the template

9.1.61. VMSSPI_MissingMember

Type:	Monitor
Description:	A disk is missing as member of a shadow set.

Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Critical

9.1.62. VMSSPI_PageFileFreeSpace

Type:	Monitor
Description:	The total free space in all page files is below a specified threshold.
Frequency:	As defined by the INTERVAL parameter
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the percentage of free space in the page file or files. Verify the additional conditions defined in this template, and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: less than 55% free space in page files. ● Major: less than 60% free space in page files. ● Minor: less than 65% free space in page files. ● Warning: less than 70% free space in page files. In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	As defined in the template

9.1.63. VMSSPI_ProcessMissing

Type:	Monitor
Description:	A process is missing on the system or clusterwide.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	On node basis or clusterwide
SPI Config:	None
Severity:	Critical

9.1.64. VMSSPI_ProcessSlots

Type:	Monitor

Description:	High amount of process slots is in use.
Frequency:	Every minute
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the average number of processes over the value of the SYSGEN parameters MAXPROCESSCNT times 100, calculated over a time period of 5 minutes. Verify the additional conditions defined in this template and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: more than 90% of the available process slots are in use. ● Major: more than 80% of the available process slots are in use. In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	As defined in the template

9.1.65. VMSSPI_ReshashtblDense

Type:	Monitor
Description:	A high amount of resources is being used.
Frequency:	Every minute
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is average number of resources on the system, compared to the value of the SYSGEN parameter RESHASHTBL, times 100. The average is calculated over a time period of 5 minutes. Verify the additional conditions defined in this template and adjust the severity levels according to your preferences. The default severity level in this template is "minor" (the percentage of resources found to the size of the resource hash table is more than 110). In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	Minor

9.1.66. VMSSPI_ReshashtblSparse

Type:	Monitor
Description:	A low amount of resources is being used; however, the value of the SYSGEN parameter RESHASHTBL is very high.
Frequency:	Every minute
Used By:	Performance module

Scope:	On node basis
SPI Config:	The value reported is an average number of resources on the system, compared to the value of the SYSGEN parameter RESHASHTBL, times 100. The average is calculated over a time period of 5 minutes. Verify the additional conditions defined in this template and adjust the severity levels according to your preferences. The default severity level in this template is "minor" (the percentage of resources found to the size of the resource hash table is more than 110). In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	Warning

9.1.67. VMSSPI_ShadowSetMissing

Type:	Monitor
Description:	A shadow set has been defined in the configuration file, but this shadow set is not available on the system.
Frequency:	As defined by the INTERVAL parameter
Used By:	System module
Scope:	Clusterwide
SPI Config:	None
Severity:	Critical

9.1.68. VMSSPI_SwapFileFreeSpace

Type:	Monitor
Description:	The total free space in all swap files is below a specified threshold.
Frequency:	As defined by the INTERVAL parameter
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the percentage of free space in the swap file or files. Verify the additional conditions defined in this template and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: less than 55% free space in swap files. ● Major: less than 60% free space in swap files. ● Minor: less than 65% free space in swap files. ● Warning: less than 70% free space in swap files. In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.

Severity:	As defined in the template

9.1.69. VMSSPI_SystemBIOrate

Type:	Monitor
Description:	The system BIO rate is high
Frequency:	Every minute
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the average number of buffered I/Os per second, for the entire system, and calculated over a period of 5 minutes. Verify the additional conditions defined in this template and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: more than 1000 BIOs per second made by the system ● Major: more than 600 BIOs per second made by the system ● Minor: more than 400 BIOs per second made by the system ● Warning: more than 200 BIOs per second made by the system In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	As defined in the template

9.1.70. VMSSPI_SystemDIOrate

Type:	Monitor
Description:	The system DIO rate is high.
Frequency:	Every minute
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the average number of direct I/Os per second, for the entire system, and calculated over a period of 5 minutes. Verify the additional conditions defined in this template and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: more than 1000 DIOs per second made by the system ● Major: more than 600 DIOs per second made by the system ● Minor: more than 400 DIOs per second made by the system ● Warning: more than 200 DIOs per second made by the system

	In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	As defined in the template

9.1.71. VMSSPI_SystemPageFaultRate

Type:	Monitor
Description:	The number of system page faults is high.
Frequency:	Every minute
Used By:	Performance module
Scope:	On node basis
SPI Config:	The value reported is the average number of system page faults per second, calculated over a period of 5 minutes. Verify the additional conditions defined in this template and adjust the severity levels according to your preferences. The default severity levels in this template are: ● Critical: more than 10 system page faults per second. ● Major: more than 5 system page faults per second. In some cases, you might need to define the severity levels per system, which requires you to create additional conditions for this template.
Severity:	As defined in the template

10. Alerting Module Configuration

As noted previously, VMSSPI provides modules for publishing alerts to various monitoring services such as Splunk, DataDog, Dynatrace, and PagerDuty. There is also a module that can be used to send alerts to Slack, and there are modules that can be used to write alerts to disk files, and to publish alerts to an MQTT message broker from which they can be consumed by interested applications. A null module is also provided that does nothing. The null module would most commonly be used in situations where alert notification via email is all that is required. The following sections briefly describe the configuration of these modules.

10.1. DataDog

In order to use the DataDog interface module to publish alerts to DataDog via HTTP, it is first necessary to create an API key that can be used by the interface module to communicate with the DataDog service. API keys can be created within the DataDog web UI by navigating to **Organization settings**, then clicking the **API keys link**. Once you have the API key (a 32-character hexadecimal string), the module can be configured in MESSAGES.TXT as follows, where *`api-key`* should be replaced with the newly created key:

```
use interface module "vmsspi$root:[lib]vmsspi$datadog_shr.exe"
"https://api.datadoghq.com/api/v1/events?api_key=api-key";
```

Alternatively, the endpoint URI may be specified using the logical name VMSSPI$DATADOG_URI. If the endpoint is specified in MESSAGES.TXT as "" (an empty string) the module will attempt to

determine the endpoint using the logical name VMSSPI$DATADOG_URI. If this logical name is not defined and the endpoint in MESSAGES.TXT is empty then VMSSPI will fail to start.

Note that for DataDog, instead of specifying an endpoint URL it is possible to instead specify DataDog email address to receive alert messages. This can be useful in situations where HTTP access to the Internet is blocked by your organisation. To send alerts to DataDog via email, follow the procedure described at https://docs.datadoghq.com/service_management/events/guides/email/ to create a dedicated email address. Be sure to specify "JSON" for the email format.

Once you have defined the dedicated email address, you can then use this in MESSAGES.TXT to define the DataDog interface as follows, specifying the appropriate email address:

```
use interface module "vmsspi$root:[lib]vmsspi$datadog_shr.exe"
"mailto:email-address";
```

Warning

It is critically important to prefix the email address with `mailto:` for the interface module to determine that alerts should be sent to DataDog via email instead of HTTP. It is also necessary to ensure that SMTP gateway details (host and port number) have been defined in MESSAGES.TXT, as this information is used by the DataDog email interface.

The logical name VMSSPI$DATADOG_DEBUG can be defined (to any value) to enable additional operational logging by the DataDog interface module. Note that this additional logging is verbose and includes detailed information of all interactions with DataDog. It is therefore recommended that this logical name should only be defined for debugging purposes and should not be routinely defined.

10.2. Dynatrace

The Dynatrace interface module uses the Dynatrace v2 API to send alerts and optionally for log ingestion. In order to use the interface it is necessary to first create a Dynatrace API v2 access token with scopes "ingest events", "ingest logs", and "ingest metrics". This is done via the **Access tokens** link under **Manage** on the Dynatrace UI main menu. Strictly speaking, only "ingest events" is mandatory, with "ingest logs" required only if event details are to be published to the Dynatrace log facility in addition to raising an alert, and the "ingest metrics" scope may be required by future VMSSPI versions[2].

Once the access token has been created, it is a simple matter of specifying the following details in MESSAGES.TXT, where *`access-token`* is the access token created as described above, and *`instance`* is the name of the Dynatrace service host. Note that *`access-token`* is 90+ characters in length, and to improve readability it is typically desirable to split the `use` statement across multiple lines like so:

```
use interface module "vmsspi$root:[lib]vmsspi$dynatrace_shr.exe"
"https://instance.live.dynatrace.com/api/v2/events/ingest
access-token";
```

If log ingestion is to be performed then the log ingestion endpoint should be added to the **`use`** command string command as shown below:

```
use interface module "vmsspi$root:[lib]vmsspi$dynatrace_shr.exe"
"https://instance.live.dynatrace.com/api/v2/events/ingest
```

[2]Given that creating and deleting access tokens is a relatively straightforward process, it may be preferable to generate a token with only those scopes required at this time and to replace this at a later date with a token having additional scope, if necessary.

```
access-token
https://instance.live.dynatrace.com/api/v2/logs/ingest";
```

The logical name VMSSPI$DYNATRACE_DEBUG can be defined (to any value) to enable additional operational logging by the Dynatrace interface module, and the logical name VMSSPI$HTTP_PROXY can be used to specify an HTTP proxy if the OpenVMS system running VMSSPI does not have direct access to the Dynatrace service endpoint(s). These logical names may be defined at the system or the process level; however it is generally recommended to define them at the system level in the file SYS$MANAGER:VMSSPI$LOGICALS.COM to ensure that they are visible to all VMSSPI processes.

It should be noted that the additional logging performed by the module when the logical name VMSSPI$DYNATRACE_DEBUG is defined is verbose and includes detailed information of all interactions with the Dynatrace service. It is therefore recommended that this logical name should only be defined for debugging purposes and should not be routinely defined.

10.3. The File Module

The file module simply logs alert details to the output file for the VMSSPI process in question, as specified in SYS$STARTUP:VMSSPI$STARTUP.COM (for details, see *Section 8, "Log Files"*). From a configuration perspective, the file module takes no parameters and is enabled in MESSAGES.TXT as follows:

```
use interface module "vmsspi$root:[lib]vmsspi$file_shr.exe" "";
```

10.4. MQTT Interface Module

The MQTT alerting module (VMSSPI$MQTT_SHR.EXE) can be used to publish alerts to an MQTT-compliant message broker, whereupon alert messages can be consumed by subscribing applications and actioned as necessary. The simplest way to configure this module in MESSAGES.TXT is to specify a **use** command as follows, where *hostname* and *port* are the host name (or TCP/IP address) of the server on which the broker is running and the port number on which the broker is listening, respectively. Note that it is possible to specify additional details in the URI connect string, including username and password details, if credentials are required in order to connect to the broker:

```
use interface module "vmsspi$root:[lib]vmsspi$mqtt_shr.exe"
"tcp://hostname:port;
```

If the address of the broker is specified in MESSAGES.TXT as "" (an empty string) the module will attempt to determine the address using the logical name VMSSPI$MQTT_URI. If this logical name is not defined and the broker address in MESSAGES.TXT is empty then VMSSPI will fail to start.

Specifying the connect string (URI) as described above is all that is required in order to start using the MQTT module; however, there are a number of optional configuration parameters for the MQTT module that can be specified using the logical names described below. These logical names may be defined at the system or process level; however it is generally recommended to define them at the system level in the file SYS$MANAGER:VMSSPI$LOGICALS.COM to ensure that they are visible to all VMSSPI processes.

Logical name	Description
VMSSPI$MQTT_URI	Can be used to specify the URI (connect string) to be used by VMSSPI processes to connect to the desired MQTT broker. As noted previously, this logical name will only be used if the URI has not been supplied in MESSAGES.TXT.

Logical name	Description
VMSSPI$MQTT_DEBUG	Defining this logical name (to any value) will cause the MQTT module to log additional operational information, including details of all alert-type publish events. In general, this logical name should only be used for debugging purposes and *should not* be routinely defined.
VMSSPI$MQTT_TOPIC_PREFIX	By default, the MQTT module creates MQTT topics that start with `vmsspi`. This logical name can be used to specify an alternative prefix. The defined value must conform to MQTT standards and should not exceed 32 bytes in length. Details of how the MQTT module constructs topics are provided below.
VMSSPI$MQTT_QOS	By default, the MQTT module uses MQTT QoS (quality of service) level 1. This logical name can be used to specify a different quality of service, with permitted values being 0, 1, and 2 (other values will cause an error). In general, it is not recommended to change the quality of service value, with level 1 being suitable for most VMSSPI use cases.
VMSSPI$MQTT_HEARTBEAT	MQTT connections are generally long-running, and in order to prevent firewalls and other such network components from dropping connections between VMSSPI processes and the MQTT broker when alerts are infrequent, the VMSSPI MQTT module implements a heartbeat thread that periodically pings the broker in order to keep the connection alive. The default value is 20 seconds; however, this can be overridden by defining an alternative value using this logical name. In general, the default value will be perfectly acceptable.

Topics created by the MQTT module when publishing alerts are used the following form, where *`org-id`* is the organization code as specified in MESSAGES.TXT, *`node-name`* is the OpenVMS node name of the server on which VMSSPI is running, and *`severity`* is the severity level of the alert (`critical`, `minor`, `major`, `normal`, `unknown`, `warning`, or `none`):

```
vmsspi/org-id/node-name/severity
```

As noted above, an alternative topic prefix (other than `vmsspi`) may be defined using the logical name VMSSPI$MQTT_TOPIC_PREFIX.

Using this scheme for topic creation, it is readily possible for consumers to consume exactly the messages that they are interested in. For example, a particular consumer may only be interested in critical alerts raised by a particular node.

10.5. The NULL Module

As its name suggests, the null alerting module (VMSSPI$NULL_SHR.EXE) does nothing, and is generally used in situations where it is only necessary to receive alert notifications via email. This is also the default module configured in VMSSPI$DATA:MESSAGES.TXT when VMSSPI is first installed.

From a configuration perspective, the null module takes no parameters, and is enabled as follows in MESSAGES.TXT:

```
use interface module "vmsspi$root:[lib]vmsspi$null_shr.exe" "";
```

10.6. PagerDuty

From the VMSSPI perspective, configuring the PagerDuty interface is straightforward; however before this can be done it is first necessary to define a new service in PagerDuty, to add an integration to this service, and to create an API access key that can be used by VMSSPI to publish alerts to the newly defined service. Other configuration work will typically also be required to define how PagerDuty should handle the various alerts that it can receive from VMSSPI. A detailed description of this configuration work is beyond the scope of this document, and the reader should refer to the online PagerDuty documentation for more detailed information; however with regard to the requirements of the VMSSPI interface module, the following points should be noted:

- The name assigned to the new service is not important; however it should be something meaningful that makes it obvious that the service relates to VMSSPI and OpenVMS.
- The new service should be able to create both alerts and incidents. VMSSPI currently uses only alerts; however this may change in the future.
- When adding an integration for the new service, the integration must use the PagerDuty Events API v2. Be sure to take a copy of the integration key, as this will be needed when configuring the VMSSPI PagerDuty interface.
- The API Access Key you create must be for the PagerDuty v2 API and the access level associated with the key must be "full".

Once PagerDuty has been configured as described and you have the integration and API access keys, the VMSSPI PagerDuty module (VMSSPI$PAGERDUTY_SHR.EXE) can be configured as follows, where `https://events.pagerduty.com/v2/enqueue` is the endpoint for the PagerDuty v2 API, *`integration-key`* and *`access-key`* should be replaced with the integration key and access key values, respectively. Note that access and integration keys are many characters in length, and to improve readability it is typically desirable to split the `use` statement across multiple lines as shown:

```
use interface module "vmsspi$root:[lib]vmsspi$pagerduty_shr.exe"
"https://events.pagerduty.com/v2/enqueue
access-key
integration-key";
```

The logical name VMSSPI$PAGERDUTY_DEBUG can be defined (to any value) to enable additional operational logging by the PagerDuty module, and the logical name VMSSPI$HTTP_PROXY can be used to specify an HTTP proxy if the OpenVMS system running VMSSPI does not have direct access to the PagerDuty service. These logical names may be defined at the system or the process level; however it is generally recommended to define them at the system level in the file SYS$MANAGER:VMSSPI$LOGICALS.COM to ensure that they are visible to all VMSSPI processes.

It should be noted that the additional operational logging performed by the module when the logical name VMSSPI$PAGERDUTY_DEBUG is defined is verbose and includes detailed information of all interactions with the PagerDuty service. It is therefore recommended that this logical name should only be defined for debugging purposes and should not be routinely defined.

10.7. Slack Integration

The Slack interface module uses the Slack Incoming Webhook mechanism to publish messages directly into a nominated Slack channel. In order to use this facility, your Slack administrator will need to define a Slack application (if you do not already have a suitable one), ensure that web hooks are enable, and create a web hook that can then be used by the VMSSPI Slack interface. A detailed description of this process can be found at https://api.slack.com/messaging/webhooks. Once you have the web hook URI,

the VMSSPI Slack module can be configured as shown below, using your web hook URI in place of the dummy value shown:

```
use interface module "vmsspi$root:[lib]vmsspi$slack_shr.exe"
"https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX";
```

If the web hook URI is "" (an empty string) the module will attempt to determine the web hook URI details using the logical name VMSSPI$SLACK_URI. If this logical name is not defined and the web hook address in messages.txt is empty then VMSSPI will fail to start. Additionally, the logical name VMSSPI$SLACK_DEBUG can be defined (to any value) to enable additional operational logging by the interface module, and the logical name VMSSPI$HTTP_PROXY can be used to specify an HTTP proxy if the OpenVMS system running VMSSPI does not have direct access to the web hook address. These logical names may be defined at the system or the process level; however it is generally recommended to define them at the system level in the file SYS$MANAGER:VMSSPI$LOGICALS.COM to ensure that they are visible to all VMSSPI processes.

Note that your Slack environment may impose various rate limits that could impact the reliable receipt of all alerts (keep in mind that Slack is primarily a communication tool for humans). For this reason, it is recommended that the Slack interface only be used for testing purposes or for situations where there are likely to be few alerts and alerts are generally not of a critical nature. Future versions of VMSSPI may look to use more advanced integration features provided by Slack that do not have such limitations.

10.8. Splunk

The VMSSPI Splunk interface module (VMSSPI$SPLUNK_SHR.EXE) uses the Splunk HEC (HTTP Event Collector) interface to send alerts into Splunk via HTTPS. Assuming that HEC is enabled within your Splunk environment, in order to use the interface it is first necessary to create a HEC token via the Splunk UI. Details for creating and managing HEC tokens can be found at the following URL. It should also be noted that HEC can only be used with the Splunk Cloud Platform and with Splunk Enterprise, as described here: https://docs.splunk.com/Documentation/Splunk/latest/Data/UsetheHTTPEventCollector.

Once HEC has been enabled within Splunk environment and you have a HEC token, it is simply a matter of specifying the following details in MESSAGES.TXT, where *instance* is the name of the Splunk service host (assuming a Splunk cloud deployment) and *hec-token* is your Splunk HEC token. Note that for Splunk Enterprise deployments, the endpoint specification may be somewhat different.

```
use interface module "vmsspi$root:[lib]vmsspi$splunk_shr.exe"
"https://instance.splunkcloud.com:8088/services/collector/event hec-token";
```

The logical name VMSSPI$SPLUNK_DEBUG can be defined (to any value) to enable additional operational logging by the Splunk module, and the logical name VMSSPI$HTTP_PROXY can be used to specify an HTTP proxy if the OpenVMS system running VMSSPI does not have direct access to the Splunk HEC service endpoint. These logical names may be defined at the system or the process level; however it is generally recommended to define them at the system level in the file SYS$MANAGER:VMSSPI$LOGICALS.COM to ensure that they are visible to all VMSSPI processes.

It should be noted that the additional operational logging performed by the module when the logical name VMSSPI$SPLUNK_DEBUG is defined is verbose and includes detailed information of all interactions with the Splunk HEC service. It is therefore recommended that this logical name should only be defined for debugging purposes and should not be routinely defined.

10.9. Syslog Interface

The VMSSPI$SYSLOG_SHR.EXE module can be used by VMSSPI to send alerts to a central Linux syslog daemon, whereupon the data can be processed as required by tools and services that cannot be

used directly within the VSI OpenVMS environment. From the VMSSPI perspective, configuration of the VMSSPI$SYSLOG_SHR.EXE module is straightforward, involving the specification of the following `use` statement in VMSSPI$DATA:MESSAGES.TXT, where *`hostname`* and *`port`* are the host name (or TCP/IP address) of the target Linux system and the port number used by the syslog daemon on that system, respectively. The last two parameters (*`sd-id`* and *`facility`*) are optional, but it should be noted that it is not possible to specify a *`facility`* without also specifying a *`sd-id`* value. It should be noted that messages will be sent to the central syslog daemon using UDP (not TCP), and the syslog daemon should be configured accordingly.

```
use interface module "vmsspi$root:[lib]vmsspi$syslog_shr.exe" "hostname port sd-
id facility";
```

The *`sd-id`* value specifies a structured data element ID for an RFC 5424 message header. The ID (a name plus possibly @digits) is case-sensitive and uniquely identifies the type and purpose of the element. If no value is specified, VMSSPI will use the default, which is `vmsspi@32473`; however it is important to note that the value 32473 has been reserved by IANA (the Internet Assigned Numbers Authority) to be used as an example number in documentation, and it is therefore suggested to use another value that is unique within your syslog environment.

The *`facility`* value can be used to specify how the message is logged by syslog, and is prepended to the message severity to effectively define the priority to be associated with the message. By default VMSSPI uses a value of `user` for the facility, and severity may be `crit`, `err`, `alert`, `info`, or `warning`, derived from the VMSSPI message severity level.

For more information regarding the specification and function of the *`sd-id`* and *`facility`* parameters, see https://www.rfc-editor.org/rfc/rfc5424.

In addition to the actual message text, VMSSPI will also include in the data sent to syslog the RFC 5424 structured data element parameters (name-value pairs) listed in the table below. These parameter names and their values can then be used by tools consuming syslog data (or by syslog itself) for indexing purposes, allowing data to be readily searched or filtered as required.

Name	**Description**
host	Specifies the OpenVMS host name
msg_name	Specifies the name of the VMSSPI message
facility	Always set to "VMSSPI"
classification	Will be set to "performance", "availability", "security", "operational", or "none" as specified in MESSAGES.TXT for the message in question

Note that the VMSSPI$SYSLOG_SHR.EXE module transmits messages to the central Linux-based syslog facility using an OpenVMS implementation of the syslog logger utility, which resides in the directory VMSSPI$ROOT:[BIN]. This is a faithful port of the Linux version, not restricted to being used within the context of VMSSPI. For example, it may be used to transmit to the remote syslog daemon other data such as Apache web server log files. For such operations, a foreign command may be defined as follows to run the logger:

```
$ logger :== $vmsspi$root:[bin]logger.exe
```

Defining the logical name VMSSPI$SYSLOG_DEBUG (to any value) will cause the VMSSPI syslog module to log additional operational details, including details of all logger commands. This logical name can be defined at the process or system level; however it is generally most appropriate to define it in the file SYS$MANAGER:VMSSPI$LOGICALS.COM as a system logical name to ensure that it will

be visible to all VMSSPI processes. In general, this logical name should only be defined to help debug problems associated with the syslog interface. Having it always defined may result in large and "noisy" VMSSPI log files, depending upon the number of alerts raised and the frequency with which logs are cycled and purged.

10.10. RabbitMQ Interface Module

The RabbitMQ alerting module (VMSSPI$RABBITMQ_SHR.EXE) can be used to publish alert messages to a RabbitMQ topic exchange, whereupon the alert messages can be consumed by any subscribing applications and actioned as necessary. The simplest way to configure this module in MESSAGES.TXT is to specify a **use** command as follows, where *URI* is the connect string for the target RabbitMQ message broker. Note that it is possible to specify various details in the URI connect string, including username, password, vhost, and port number. It should also be noted that the URI scheme can be "amqp" for non-encrypted communication or "amqps" if OpenSSL support is enabled for the broker. It is assumed that no client certificate is required (this assumption may be eliminated in future releases):

```
use interface module "vmsspi$root:[lib]vmsspi$rabbitmq_shr.exe" "URI";
```

If the address of the broker is not specified, the module will attempt to determine the address using the logical name VMSSPI$RABBITMQ_URI. If this logical name is not defined and the broker address in MESSAGES.TXT is empty, then VMSSPI will fail to start.

Specifying the connect string (URI) as described above is all that is required in order to start using the RabbitMQ module. However, there are a number of optional configuration parameters for the module that can be specified using the logical names described in the table below. These logical names may be defined at the system or process level; however, it is generally recommended to define them at the system level in the file SYS$MANAGER:VMSSPI$LOGICALS.COM to ensure that they are visible to all VMSSPI processes.

Logical Name	Description
VMSSPI$RABBITMQ_URI	Can be used to specify the URI (connect string) to be used by VMSSPI processes to connect to the desired RabbitMQ message broker. As noted previously, this logical name will only be checked for and used if the URI has not been supplied in MESSAGES.TXT.
VMSSPI$RABBITMQ_DEBUG	Defining this logical name (to any value) will cause the RabbitMQ module to log additional operational information, including details of all published alerts. In general, this logical name should only be used for debugging purposes and should not be routinely defined.
VMSSPI$RABBIT_TOPIC_PREFIX	By default, the RabbitMQ module constructs routing keys that start with "vmsspi" (see details below). This logical name can be used to specify an alternative prefix. The defined value must conform to RabbitMQ standards for routing key naming and should not exceed 32 bytes in length.
VMSSPI$RABBITMQ_EXCHANGE	This logical name can be used to specify the name of the RabbitMQ exchange to which alerts will

Logical Name	Description
	be published. By default (if this logical name is not defined), alerts will be published to the default RabbitMQ topic exchange, "amq.topic". Whilst not a strict requirement, it will generally be desirable for alerts to be published to a topic exchange, as this provides the greatest flexibility, allowing consumers of alerts to consume only those alerts they are interested in, such as those for a particular node or a particular severity (or possibly a combination thereof).

The RabbitMQ module publishes alerts to the nominated exchange with routing keys of the form shown below, where *org-id* is the organization code as specified in MESSAGES.TXT, *node-name* is the OpenVMS node name of the server on which VMSSPI is running, and *severity* is the severity level of the alert (`critical`, `minor`, `major`, `normal`, `unknown`, `warning`, or `none`):

`vmsspi.`*`org-id`*`.`*`node-name`*`.`*`severity`*

As noted above, an alternative topic prefix (other than `vmsspi`) may be defined using the logical name VMSSPI$RABBITMQ_TOPIC_PREFIX.

Using this scheme for routing keys, it is readily possible for consumers to consume only the alerts they are interested in. For example, a particular consumer may only be interested in `critical` alerts, regardless of *org-id* and *node-name*, which could be facilitated by consuming from a queue bound to the relevant exchange with the following biding pattern:

```
vmsspi.*.*.critical
```