VSI OpenVMS

PERFDAT V4.8

Architecture and Technical Description

February 2019

Revision/Update Information	New Manual
Software Version	VSI PERFDAT V4.8
Operating System Version	OpenVMS Alpha V7.3-2 & higher OpenVMS I64 V8.2 & higher

February 2019

Contents

Preface

This manual includes:

- Description of the VSI PERFDAT architecture
- Description of the VSI PERFDAT software components
- Installation and configuration guide

Audience

This manual provides an overview of the overall architecture, the features of VSI PERFDAT, the software components involved, their interactions as well as a description of how to install and configure VSI PERFDAT. The reader should be familiar with

- The OpenVMS Operating System V7.3-2 or later
- TCPIP Services for OpenVMS

Document Structure

- Chapter 1 Introduction to VSI PERFDAT
- Chapter 2 VSI PERFDAT architecture
- Chapter 3 Technical Description of the software components
- Chapter 4 Directory structure and Logicals
- Chapter 5 Tools
- Chapter 6 Appendix A: Available statistics

Conventions Used in this Manual

Special	in examples indicates text that the system displays or user type input.
UPCASE	in a command represents text that you have to enter as shown.
Lowercase Italics	indicates variable information that a user supplies.
[ ]	in a command definition, enclose parts of the command that a user can omit.
Key	indicates a named key on the keyboard; for example, RETURN
CTRL/x	is the symbol used to represent the pressing of a control key. It indicates that the user holds down the key marked Ctrl and press the appropriate key.

1

Introduction to VSI PERFDAT

This chapter provides a brief introduction to VSI PERFDAT for OpenVMS. For a more detailed description of the components and functions of VSI PERFDAT, see chapter VSI PERFDAT software architecture and chapter VSI PERFDAT technical description.

What is VSI PERFDAT?

VSI PERFDAT is a performance and capacity planning solution that is capable of supporting all performance and capacity planning related activities during the lifetime of a system.

It is supported on:

- HP OpenVMS V7.3-2 Alpha
- HP OpenVMS V8.2 Alpha
- HP OpenVMS V8.3 Alpha
- HP OpenVMS V8.4 Alpha
- HP OpenVMS V8.2 I64
- HP OpenVMS V8.2-1 I64
- HP OpenVMS V8.3 I64
- HP OpenVMS V8.3-1H1 I64
- HP OpenVMS V8.4 I64
- VSI OpenVMS V8.4-1H1 I64
- VSI OpenVMS V8.4-2 I64
- VSI OpenVMS V8.4-2L1 I64

VSI PERFDAT is an OpenVMS solution to collect performance data from one OpenVMS systems or however many systems exist in your environment. In addition VSI PERFEDAT provides two extensions:

- VSI PERFDAT EVA extension
- VSI PERFDAT SNMP extension

These extensions enable the user to collect performance data from non-OpenVMS systems.

The VSI PERFDAT EVA extension collects performance data from HP StorageWorks Enterprise Virtual Arrays (EVA).

With the VSI PERFDAT SNMP extension a user is able to monitor any system that provides performance data via SNMP like Tru64 systems, Solaris systems, Linux systems or Brocade switches.

VSI PERFDAT has been designed to work with standard OpenVMS components. This means that the performance database used is based on RMS. This database is a high speed, write optimized, distributed and post relational database.

Data from the performance database are accessed via the common DQL interface. The DQL interface includes a data and a network abstraction layer. Due to the data abstraction layer data access is independent of the record format of the database. The network abstraction layer provides the ability to access data located on any remote node transparently.

For data visualization and analysis a PC based GUI is provided. The VSI PerfDat GUI is supported on Windows 2000/2003/2008/2012/XP/7/ 8.1 and Windows 10.

Key design goal

From the very beginning VSI PERFDAT was designed as a powerful solution that is capable to support all performance and capacity planning related activities during the lifetime of a system. These include:

- Benchmarking
- Stress testing
- System sizing
- System characterization
- Tuning
- Troubleshooting
- Online Monitoring
- Investigation of performance anomalies
- Validating the performance impact of new software/software versions/OpenVMS releases
- Trend and Capacity Analysis

In addition the VSI PERFDAT solution was designed for the performance management of large distributed environments with systems installed all over the world. A key goal was to provide a solution that requires no manual data management in order to analyze the data. Thus, data has to be transparently accessible via a single access point regardless of the data storage location.

Key requirements

In order to achieve the design goals several requirements have to be fulfilled

- Easy to manage and control (plug and play)
- The ability to manage huge amounts of data (> 1TByte)
- As little data management as necessary
- Best practice workflow support based on a variety of statistical functions for any kind of performance analysis task in order to

    - Reduce analysis time
    - Receive feedback about what is going on without expert knowledge
- Analysis tool that does not depend on the source data format adhering to the principle of “Analyze what you get”
- Data analysis without data pre-processing
- Automatic trend reporting and data compression
- Archive and housekeeping functionality
- Data from different sources (different systems – native data of the VSI PERFDAT data collector, mapped or imported data) can be transparently accessed via one single common interface
- Data analysis depends neither explicitly nor implicitly on the start time nor on the sample interval of any data collection
- Easy data transfer of the performance data base, or parts of it, for offline analysis
- Up- and backward data compatibility
- Ability to map/import data from additional data sources
- State of the art GUI
- No dependency on any layered product except those available on the OpenVMS installation CD
- No dependency on any 3rd party product or any kind of shareware/freeware

VSI PERFDAT meets all these requirements.

2

VSI PERFDAT software architecture

This chapter provides an overview of the software architecture and the interactions of the SW-components of VSI PERFDAT. For a more detailed description of the VSI PERFDAT SW-components, see chapter VSI PERFDAT technical description.

Architectural overview

The components of the VSI PERFDAT Performance and capacity planning solution software are listed below

- OpenVMS Data Collector (up to 3 collections can be run simultaneously)
- VSI PERFDAT EVA extension – with the EVA extension performance data from up to 64 HP StorageWorks Virtual Arrays (EVA) can be collected by any single OpenVMS node.
- VSI PERFDAT SNMP extension – with the SNMP extension performance data from up to 64 non OpenVMS systems providing performance data via SNMP can be collected by any single OpenVMS node.
- Application programming interface
- Distributed performance database
- VSI PERFDAT configuration database
- VSI PERFDAT cluster view database
- Data Query Interface (DQL)
- Performance database file name cache service
- Online performance alerting
- Statistics package
- Auto trend engine
- Graphical User Interface
- Auto Archiving and housekeeping
- Management Interface (PERFDAT_MGR)

Fig. 1.1 shows the overall software architecture of the VSI PERFDAT solution.

The OpenVMS data collector, the EVA extension and the SNMP extension collect performance raw-data from the appropriate system according to the definitions of the collection profiles the performance data collections were started with. The raw-data is converted into human readable format and inserted directly into the distributed performance database.

Collection profiles are stored in the collection profile table of the VSI PERFDAT configuration database. These collection profiles are managed and maintained by the management utility PERFDAT_MGR.

The sum of all data files that exist within a monitored environment, regardless if all systems in the environment have direct access to all or a part of these files is called the distributed performance database.

Apart from the OpenVMS data collector, the EVA extension and the SNMP extension the auto-archiving process accesses the data files of the distributed performance database directly. All other SW-components, as shown in Fig. 1.1, access data via the common DQL interface. The DQL interface provides a data abstraction and a network abstraction layer.

The data abstraction layer provides the feature that the data access is independent of the record format of the physical storage area of the performance database and guarantees forward- and backward data compatibility.

The network abstraction layer grants transparent access to any data within the defined community. A community is a logical partition of the whole environment and defines the database view when accessing the data via one system within a community regardless of where the data files are actually stored within the community. Systems of a particular interest to a VSI PERFDAT user can be configured in the context of a community. The systems that belong to a particular community is freely definable e.g. all members of a cluster might be part of a particular community, or standalone systems running the same application may be part of another community. The community definition is not cluster bound.

The DQL interface provides the feature of dynamic CSV file mapping. Dynamic CSV file mapping means that the CSV files can be accessed via the DQL interface as if these files were part of the distributed performance database. CSV data is processed by the archiving process, and some statistical methods are not applicable.

CSV data can be loaded and/or imported into an existing collection database. The main difference between CSV load and import is that CSV data is normalized before being inserted into the collection database. It is very likely that the time stamps in the CSV files do not match the time stamps in the collection database. Normalizing means that based on the CSV data expectancy values are calculated for the time stamps of the collection database. An integral based algorithm is used to normalize the data.

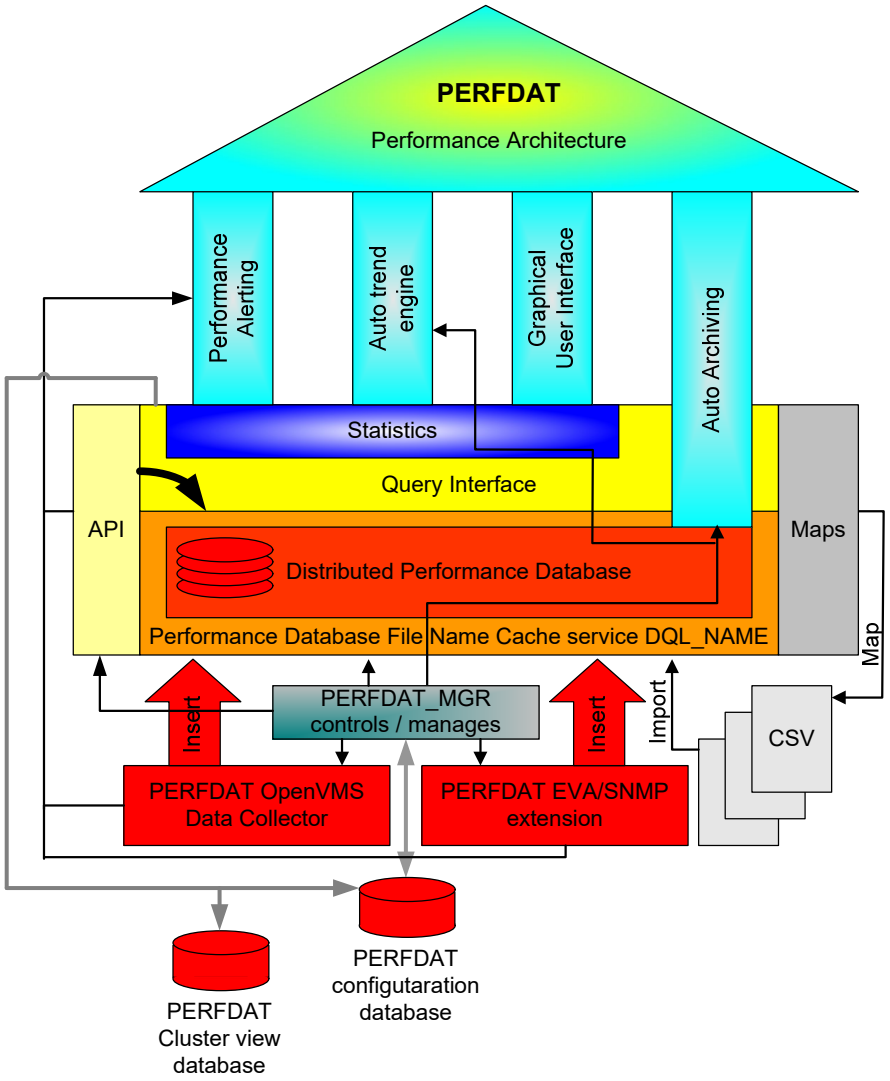


Fig. 1.1 VSI PERFDAT software architecture

In addition VSI PERFDAT provides an easy to use C programming interface (API) to insert any type of performance data collected by the components (programs) of an application directly into the distributed VSI PERFDAT performance database. Performance data collections created by use of the VSI PERFDAT API are called application data collections. One of the most important features of the VSI PERFDAT API is that application data collections can be monitored and controlled as any other data collection created by the OpenVMS data collector the SNMP extension or the EVA extension without any programming effort, code change or the need of restarting the application.

The DQL interface provides the features to create cluster views and to define side specific measures.

A cluster view maps performance data of different nodes for cluster wide performance analysis. Once a cluster view is created a virtual collection database is accessible. The advantage is that such a virtual cluster view collection database can be accessed in the same way by the DQL$ utility and the VSI PERFDAT GUI as if it is a collection database created by the OpenVMS data collector, the EVA extension or the SNMP extension. Thus, all methods and

features to analyse performance data of single nodes are available for cluster views too. Consequently the workflow for cluster analysis does not differ from the workflow to analyse single node performance data.

Although in most cases cluster views will be created for cluster wide performance data analysis of OpenVMS clusters there exists no restriction that performance collection databases of OpenVMS cluster members only can be members of a cluster view. Any collection database of any node available can be added to a cluster view. The only restriction is that all collection databases of a cluster view were created with the same sample interval.

User defined statistics are calculated values that can, once defined, be accessed as if they are part of the of the collection databases available.

The statistic package is part of the DQL interface. The statistic package provides all the statistical methods (such as correlation, deviation analysis etc.) for advanced data analysis.

The PC based graphical user interface and the auto trend engine access data via the DQL interface.

The performance database file name cache service DQL_NAME provides a database file name cache to all VSI PERFDAT components that contains full header information about all PERFDAT database files locally stored.

Within the VSI PERFDAT environments archive systems can be defined. The role of an archive system is to store and manage raw-data of the systems that are assigned to that archive system. The main task of the auto archiving process on a collecting system is to move the performance data files periodically to that archive system and to clean up log- and temp-files. (For more detailed information see chapter VSI PERFDAT technical description)

The auto trend engine automatically creates trend and capacity reports from the raw-data collected by the OpenVMS data collector, the EVA extension and the SNMP extension. The content of such reports are defined by so called trend report profiles that are stored in the trend report table of the VSI PERFDAT configuration database. These trend report profiles are managed and maintained by the management utility PERFDAT_MGR, and are never touched by the archiving process.

Part of the DQL interface is the DCL callable DQL$ utility. This utility is the OpenVMS data query and data content management interface (e.g. creating physical storage areas, creating tables, dropping tables etc.)

The PERFDAT_MGR utility is the interface for managing the VSI PERFDAT environment. This includes controlling and monitoring data collections, controlling the auto trend engine, the auto archiving process, the online alerting subsystem, the performance database file name cache service as well as managing and maintaining the VSI PERFDAT configuration database.

The online performance alerting subsystem provides real-time alerting capabilities. Online performance alerting can be enabled for any active performance data collection, independently if the data collection is performed by the OpenVMS data collector, the EVA extension or the SNMP extension.

Once online alerting has been enabled for an active performance data collection the alerting subsystem tracks the actual values of specific statistics collected by the OpenVMS data collector, the EVA extension and the SNMP extension and triggers alerts if any alert condition becomes true.

The statistics to monitor, the alert conditions and the alert method (OPCOM messages, user definable command scripts) are defined by alert blocks within an alert definition file. An alert definition file is a text file for easy customization.

VSI PERFDAT Environment

The VSI PERFDAT environment consists of so called communities (see Fig. 1.2). A community is a logical partition of the whole environment and defines the database view when accessing the data via any system within a community. All systems of particular interest can be configured within the context of a community. As stated in the previous chapter, no rules exist that limit the configuration of such communities (such as cluster boundaries, location of the systems etc.). The number of possible communities ranges from one to the total number of systems within the whole environment. Fig. 1.2 shows an example of partitioning the environment into communities and the role of the systems within the communities.
The role of the systems within a community is defined by the SW-components running on the systems.

- OpenVMS collector system
- EVA agent system (collects data from HP StorageWorks Virtual Arrays)
- SNMP agent system (collects data from SNMP server systems)
- Archive system
- Access server
- SNMP server system (provides performance data via SNMP)

OpenVMS collector system

A system is an OpenVMS collector system if the VSI PERFDAT OpenVMS data collector is running on that system. For more details see chapter VSI PERFDAT OpenVMS Data Collector.

EVA agent

A system is anEVA agent node if the VSI PERFDAT EVA extension is running on that system and HP StorageWorks Virtual Array (EVA) performance data collections are active. For more details see chapter VSI PERFDAT EVA extension.

SNMP agent

A system is a SNMP agent node if the VSI PERFDAT SNMP extension is running on that system and data collections are active for remote systems providing performance data via SNMP (such as Tru64, Brocade switches, etc.). For more details see chapter VSI PERFDAT SNMP extension.

Archive system

A system is called an archive system, if VSI PERFDAT OpenVMS collector systems and/or SNMP agent systems are configured to move PERFDAT collection data files to this system periodically (i.e. auto archiving is enabled on the collecting system).
For more details see chapter VSI PERFDAT Archiving and Housekeeping.

Access server

A system is called an access server, if the VSI PERFDAT DQL interface is configured and started. For more details see chapter VSI PERFDAT Query Interface (DQL).

SNMP server system

An SNMP server system is any non-OpenVMS system that provides performance data via SNMP and is supported by the SNMP extension. Currently Tru64 Unix, Solaris, Linux and Brocade Switches are supported. Other operating systems and components will be supported in future releases of VSI PERFDAT.

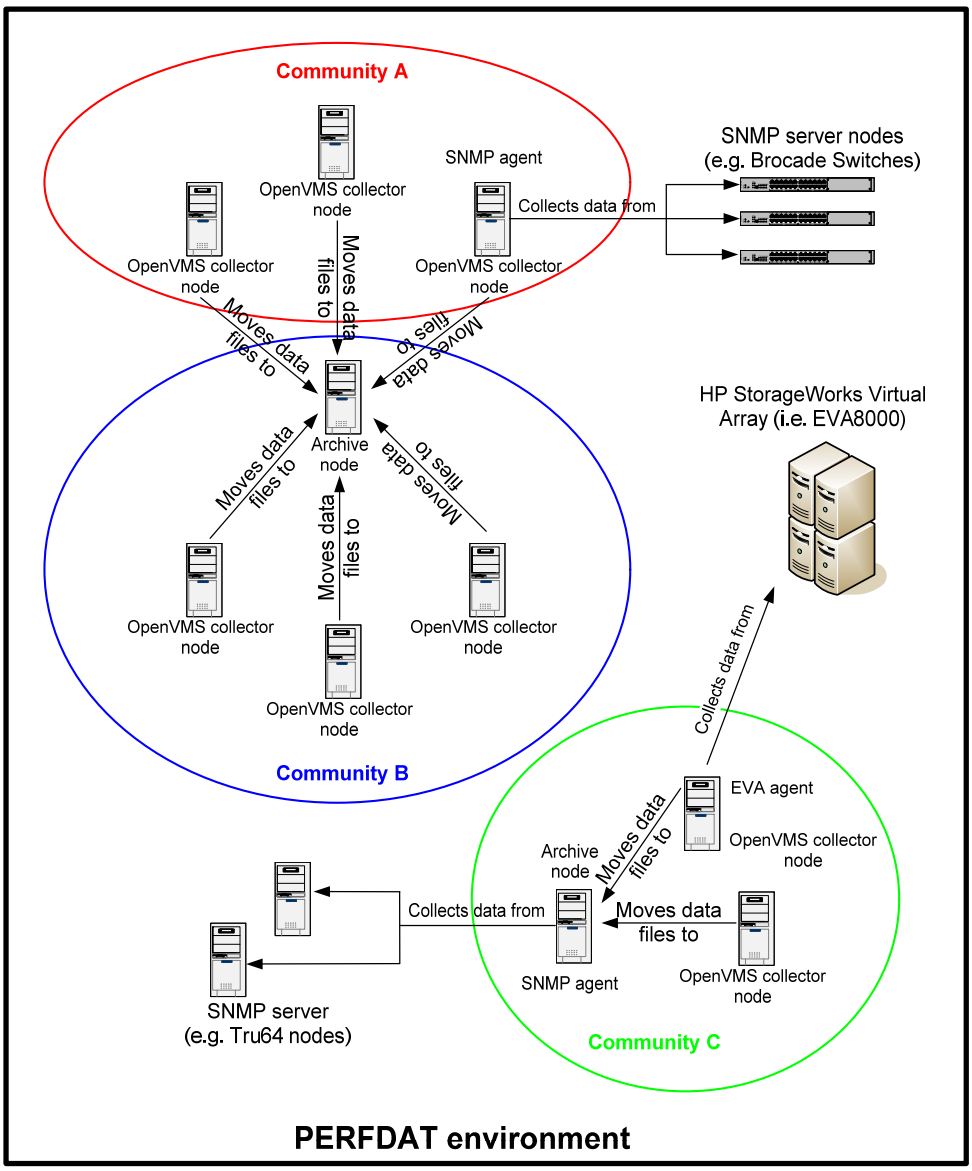


Fig. 1.2: VSI PERFDAT environment example.

As shown in Fig. 1.2, an OpenVMS system configured within the VSI PERFDAT environment can play several roles – it can be an OpenVMS collector system, an EVA agent system, a SNMP agent system, an archive system and (not shown in Fig. 1.2) an access server.

Communities within the VSI PERFDAT environment are defined by the logical PERFDAT$COMMUNITY on each system within the VSI PERFDAT environment.

PERFDAT$COMMUNITY is a system-wide logical that refers all systems of the community the local node is member of. (Assign these nodes as a comma separated list.)

Note

The logical PERFDAT$COMMUNITY has to be defined on all nodes of a community, otherwise the user will get different views on the performance database depending on the node he connects to when accessing the distributed performance database.

Data that belongs to the access server (e.g. created by the access server) is always visible when the user connects to the distributed performance database regardless of the definition of the logical PERFDAT$COMMUNITY and even if the logical is not defined.

In chapterVSI PERFDAT Query Interface (DQL) the effect of community settings on the database view is described in more detail.

The main reason for defining communities is to have a selective view on the data when accessing the performance database via an access server. E.g. if you have configured your environment as shown in Fig. 1.2 and you access the performance database via any node of community C, all data collected by these nodes are visible, regardless of where the data is actually stored within the community.

The system wide logical PERFDAT$ARCHIVE_NODE defines the archive system for each node within the environment. It contains the node name of the archive node. If that logical is de-assigned or set to “NOTDEF”, data will be stored locally. This logical can, but do does not have to be defined on the archive node.

Note

If you define PERFDAT$ARCHIVE_NODE on the archive node, please make sure that the node name assigned is the local node name. Otherwise the archiving process running on the archive node tries to move all collection data files to that node. If this action succeeds collecting nodes will loose access to this data since data access routing is not supported by the DQL interface on archive nodes.

Accessing data via a collecting node provides a community specific view of the distributed performance database. This behavior changes if the logical PERFDAT$ARCHIVE_NODE defined on the archive node refers to its own node name and the archive node is used for data access. In this case, access is granted to all data locally stored regardless if the data files were created by any community member or not, plus the data stored on the other community member nodes. ´

Example:
Looking at Fig 1.2, there is one node that is the archive node for community A and B, but is a member of community B only. If you access data via that node the performance database view consists of all data of community A and all data of community B that has already been moved to the archive node.
Consequently, even if the archive node is not a member of any community the user can access all data locally stored on the archive node when accessing the archive node.

If the archive node is not member of a community this does not imply that the user looses access to the data already moved to the archive node when accessing a community member. The archive node defined on the collecting node will always be queried for community data regardless if it is part of the community or not.

Please refer to chapter VSI PERFDAT Query Interface (DQL) to obtain more detailed information about communities and data access.

There are several reasons for defining archive nodes

- Centralized data storage – single backup and restore location
- The statistic package provides several methods for advanced data analysis that reduces analysis time and provides the ability to identify performance bottlenecks without expert knowledge. These methods are very powerful, but running these methods may cause heavy I/O load on the system. Thus, if the data is stored on a productive system, analysis runs can increase I/O load significantly and overall system performance may suffer. If data is stored on an archive node the data analysis can be done without negatively influencing the productive systems.
- The auto-trend engine extracts trend and capacity reports from performance raw-data. It is triggered once a day on any collecting node within your environment. Depending on the number of reports to be created automatically and their definition the auto trend engine may also stress the I/O subsystem. If an archive node is in use, only raw-data already stored on that node will be accessed by the auto trend engine. Thus, the auto trend engine has – as long as the archive node is up and accessible - no influence on the performance of the I/O subsystem on the collecting node.

VSI PERFDAT technical description

The following chapter provides a detailed description of the VSI PERFDAT software components and the interaction of these components.

VSI PERFDAT OpenVMS Data Collector

The VSI PERFDAT OpenVMS data collector collects performance raw-data from an OpenVMS system, converts this raw-data into human readable form, and stores the data in the distributed performance database.

Features

- Up to 3 collections in parallel
- More than 700 statistics organized in 24 metrics
- Profile controlled – profiles reside in the VSI PERFDAT configuration database and are managed via the PERFDAT_MGR utility
- Sample interval is freely definable (minimum = 1 second)
- Each of the metrics can be enabled/disabled independently
- For each of the metrics (except the system metrics), thresholds can be set to minimize the amount of data collected
- Metrics can be restricted to single/multiple devices, processes, users, images and volumes
- Device metrics allows I/O resolution to single process, files and files per process (not only hot file statistic but also the originator of hot files can be identified)
- Files in the device- and XFC metrics not only resolve to file ID's but also to their real file names
- Complete XFC integration
- Permits online monitoring
- Online performance alerting can be enabled dynamically
- Dynamic resource trimming – in order to avoid performance problems due to running VSI PERFDAT, the tool monitors its own resource consumption, and if CPU load and/or I/O load exceeds definable thresholds VSI PERFDAT automatically increases collection sample intervals and/or dismisses metrics rules.
- Controlled by PERFDAT_MGR

Available metrics

Below all the metrics available are listed. A detailed description of the statistics captured by each metrics is provided in Appendix A of this document

- System
- CPU
- Process
- User
- Image
- Account
- Device
- Device.IOTimeHist
- Device.IOSize
- Device.File
- Device.Process
- Device.Process.File
- Device.Capacity
- Device.Path (OpenVMS V7.3-1 and higher)
- IOPathes (OpenVMS V7.3-1 and higher)
- XFCVolume (OpenVMS V7.3 and higher)
- XFCVolume.IOSize (OpenVMS V7.3 and higher)
- XFCVolume.File (OpenVMS V7.3 and higher)
- XFCVolume.File.IOSize (OpenVMS V7.3 and higher)
- LANAdapter
- LANAdapter.Device
- LANProtocol
- SCSPort
- SCSPort.VC
- SCSPort.VC.Channel

General description

The VSI PERFDAT OpenVMS data collector process – its process name is VSI PERFDAT- is multithreaded in order to run up to 3 collections in parallel. PERFDAT.EXE is the data collector image and is located in the PERFDAT$BIN directory.

All informational, warning and error messages during runtime are posted to OPCOM and stored in the VSI PERFDAT OpenVMS data collector log-file. The log-file is located in the PERFDAT$LOG directory. The filename of the log-file has the format

PERFDAT_*nodename*.LOG

E.g. the file PERFDAT$LOG:PERFDAT_BCSXTC.LOG is the log-file of the VSI PERFDAT OpenVMS data collector running on node BCSXTC.

Startup / Shutdown

The VSI PERFDAT OpenVMS data collector can be started by any privileged user either via the PERFDAT_MGR utility (see section PERFDAT_MGR Management Interface):

```
$ MCR PERFDAT_MGR LAUNCH PERFDAT
```

or directly from the DCL command line by executing the interactive startup script:

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

The privileges required to start the OpenVMS data collector are

- CMKRNL
- NETMBX
- OPER
- SYSLCK
- SYSPRV
- TMPMBX
- WORLD

Note

Launching the OpenVMS data collector with either commands listed above also starts the DQL interface (DQL$SRV, PDBC$SRV), the performance database filename cache service DQL_NAME and the auto archiving process.

Fig. 2.1 shows the start-up flow-chart for the VSI PERFDAT OpenVMS data collector. The start-up sequences are identical if the user invokes either of the command scripts or if the PERFDAT_MGR management utility is used for launching the OpenVMS data collector.

VSI PERFDAT provides full multi-version support. This means that no system manager action is required to upgrade/install VSI PERFDAT if OpenVMS is upgraded to any version supported by VSI PERFDAT. The VSI PERFDAT OpenVMS data collector startup command procedure checks the actual OpenVMS version in use before starting VSI PERFDAT. If OpenVMS has been upgraded all VSI PERFDAT version specific images are replaced and loaded automatically before the VSI PERFDAT OpenVMS data collector is started to avoid version mismatch problems.

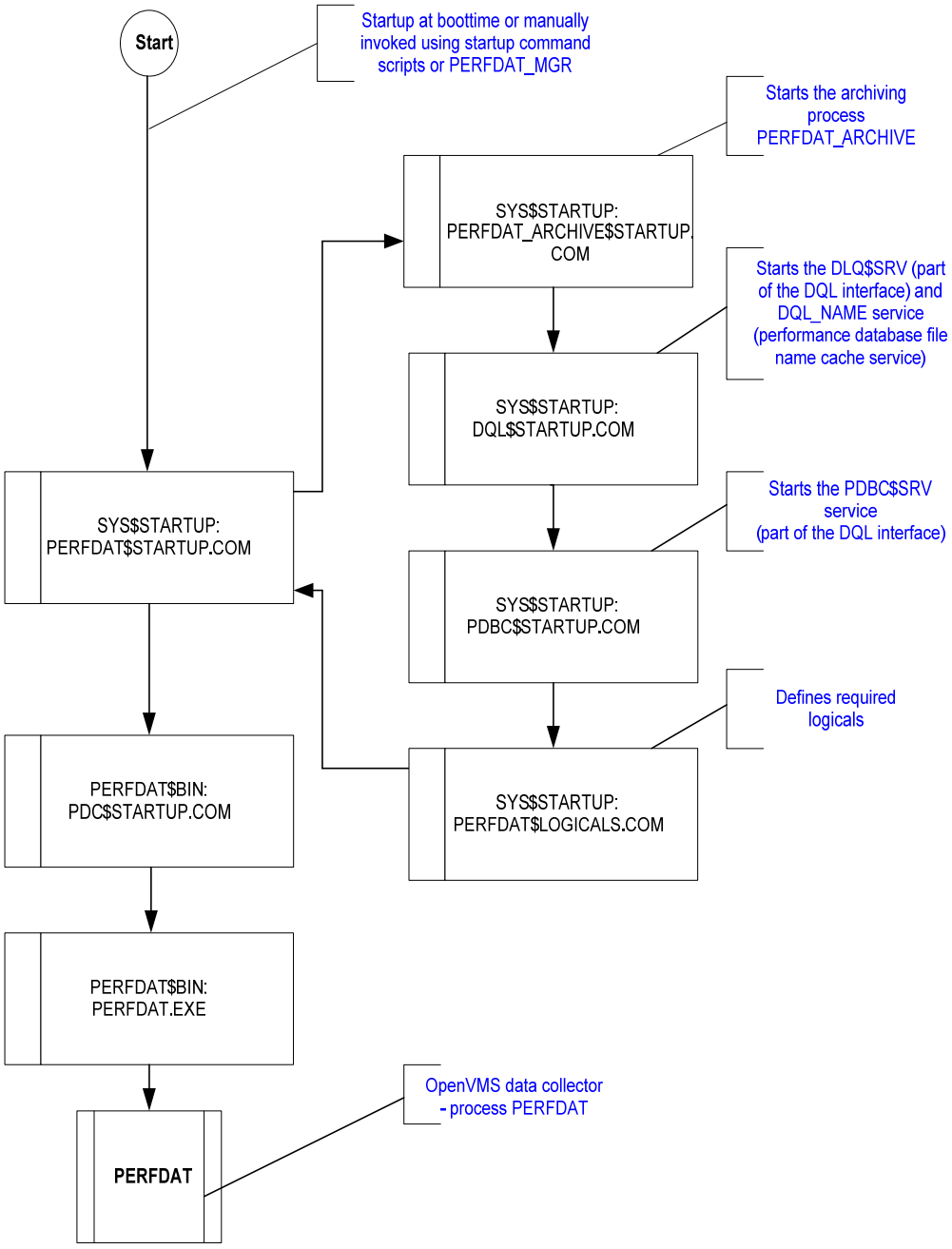


Fig. 2.1 Start-up flow-chart for the OpenVMS data collector.

In order to shutdown the VSI PERFDAT OpenVMS data collector, enter:

```
$ MCR PERFDAT_MGR SHUTDOWN PERFDAT
```

The VSI PERFDAT OpenVMS data collector will be shutdown implicitly if you shutdown the whole VSI PERFDAT environment with the command:

```
$ MCR PERFDAT_MGR SHUTDOWN ALL
```

For additional information about the actions performed when executing either of these commands, please refer to PERFDAT_MGR – Management Interface.

Initialization phase

Once the VSI PERFDAT OpenVMS data collector is started it performs the actions in the order listed below

- Initialize internal data structures and threads
- Create and initialize mailboxes for online communication with the management interface PERFDAT_MGR
- Fill internal caches
- Search the configuration database for an auto-start collections
- Checks if online performance alerting is enabled for this auto-start collection. If this is the case the OpenVMS data collector invokes the online performance alerting subsystem for this collection.

Two mailboxes are created and initialized for online communication with the PERFDAT_MGR management utility as shown in Fig. 2.2. The data collector receives management commands via the PERFDAT_COLL mailbox and transmits status information via the PERFDAT_MGMT mailbox.

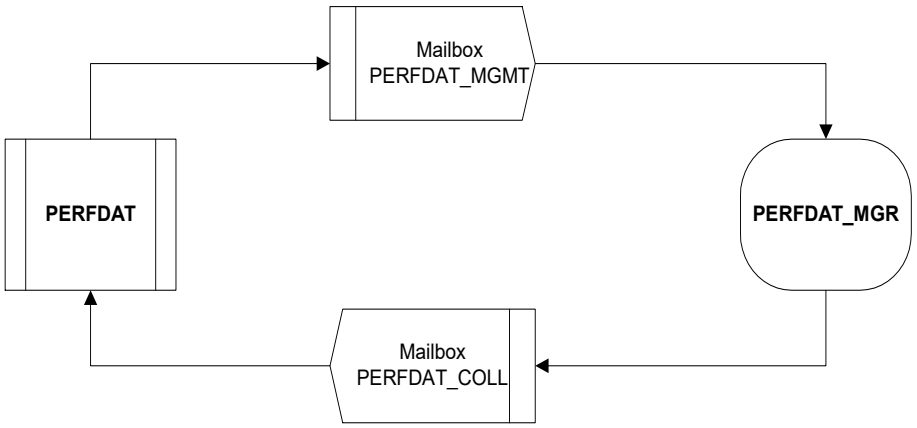


Fig. 2.2 Mailbox communication between PERFDAT_MGR and VSI PERFDAT OpenVMS data collector

The most important internal cache of the VSI PERFDAT OpenVMS data collector is its file-cache. That cache is a file ID to file name cross reference, and is required for the filename resolution if one of the metrics listed below is enabled for a collection.

- Image
- Device.File
- Device.Process.File
- XFCVolume.File
- XFCVolume.File.IOSize

In order to fill that file-cache, the VSI PERFDAT OpenVMS data collector fetches all file ID's known to the OpenVMS system (XFC & processes) and reads the corresponding file names from INDEXF.SYS.

Note

Depending on the number of files known to the OpenVMS system, the file name resolution (= filling internal file cache) can last from a few seconds up to 10 min. During that time any management requests from the PERFDAT_MGR utility are blocked. Thus, if the VSI PERFDAT OpenVMS data collector does not respond to a PERFDAT_MGR command for a few minutes after start-up it does not necessarily indicate, that the collector has crashed or is not working correctly, but that the cache filling is in progress.

Once the internal caches have been filled, the data collector checks the auto-start table of the VSI PERFDAT configuration database if there is a collection profile configured for auto-start on the local node. If the data collector finds a valid entry for the local node the associated collection is started and, if configured, online alerting is enabled for the collection without any additional user interaction. The auto-start table of the VSI PERFDAT configuration database is managed by the PERFDAT_MGR utility.

Thereafter the VSI PERFDAT OpenVMS data collector starts listening to its input mailbox (PERFDAT_COLL) for any PERFDAT_MGR management command.

Starting a collection

Data collections are profile controlled. A collection profile defines the sample interval and the metrics to collect. Collection profiles are stored in the VSI PERFDAT configuration database and are managed via the PERFDAT_MGR utility. The configuration database is:

PERFDAT$CFG:PERFDAT_PROFILES.CFG.

As has been mentioned previously, up to three collections with different profile settings can be active in parallel. Although the data collector is a single process, the collections are handled independently. Thus, no restriction exists, that the metrics have to be common to all active collections.

Data collections can be started either manually via the PERFDAT_MGR utility when the data collector is up and running or automatically after start-up (see chapter Initialization phase).

Whenever a collection is triggered via the PERFDAT_MGR utility the whole collection profile content is sent via Mailbox to the PERFDAT OpenVMS data collector (see also PERFDAT_MGR – Management Interface).

After receiving a collection start request the VSI PERFDAT OpenVMS data collector checks if the collection is already in use and if all collection threads are busy. If one of these conditions is true then the start request is rejected.

If the received start request has been accepted by the data collector, it first creates a new data collection file, accesses the record descriptor table of the VSI PERFDAT configuration database, reads the field descriptors for the enabled metrics and stores these descriptors in the header of the newly created file. The reason for copying the field descriptors into the data file is to guarantee read access to the data independent of

- Version of the DQL environment.
- Version of the VSI PERFDAT OpenVMS data collector.
- Existence of a valid VSI PERFDAT configuration database on the access server.

Afterwards the VSI PERFDAT data collector schedules the collection to start according to the settings of the collection profile. Scheduling the start request means:

- The collection is not started at the time the start request was received. Depending if another collection is already active the request collection is scheduled to start
  - the next full minute if there is no collection active
  - at the time the (active) collection with the shortest sample interval will be triggered to get the next data sample.

This is done to save system resources. If different collections are triggered at the same time to get data samples and there are metrics that are common for these collections, the data is collected only once and distributed to these collection threads.

Online alerting

Online performance alerting is performed by the online alerting subsystem and can be enabled for any active performance data collection. The online alerting subsystem is invoked by the OpenVMS data collector either on user request via the PERFDAT_MGR utility when the data collector is up and running or automatically after start-up (see chapter Initialization phase).

Data collection file

The data collector creates new data files periodically for each active collection. The rules for creating new data files are:

- Whenever a collection is started (restarted)
- Data files are closed and new data files are created daily for each active collection. The time of day the collection data files are closed is defined when a collection is started.

Thus, 1 to n data file can exist per day and collection. The VSI PERFDAT OpenVMS data collector creates all data files in the directory pointed to by the logical PERFDAT$DB_LOCAL. The file name format is:

PERFDAT_*node_yyyy-mm-dd_profile*.DAT_*sssss*

with:

- *node*
  Local node name
- *yyyy*
  Year the collection data file was created
- *mm*
  Month the collection data file was created
- *dd*
  Day the collection data file was created
- *profile*
  Collection profile used to start the performance data collection
- *sssss*
  Time since midnight when the collection data file was created

The size of the data file per collection and day depends on the metrics enabled and the number of elements collected per metric (number of processes, devices, users, files accessed per device and collection interval …). The initial file size is estimated by the data collector based on the metrics enabled and the time remaining till midnight. If a collection is running with the default collection profile provided by the installation kit the size typically ranges from 50 to 200 MB per day.

A user can restrict the initial file size by defining the system-wide logicals

- PERFDAT$DATA_MEANSIZE
  Mean record size of the data record stored to the data files.

- PERFDAT$DATA_RECORDCNT
  This logical defines the number of records that are expected to be written to a collection file.

Note

If these two logicals are defined the OpenVMS data collector will create all new collection data files with the same initial size (according to the values of these logicals). Thus, if these values are small the initial file size will be small, and it is very likely that the data files will get fragmented, and consequently the write performance to the file decreases.

Dynamic resource trimming

The VSI PERFDAT OpenVMS data collector provides a feature called dynamic resource trimming. It monitors its own system resource consumption and modifies the collections running if the resource consumption is above the defined thresholds. The system resources monitored are

- CPU load
- I/O load

CPU load triggered dynamic resource trimming

If the average CPU load of the data collector within a sample interval is above the defined threshold the data collector selectively disables the most CPU consuming metrics for each collection running. These are:

- XFCVolume.File.IOSize
- XFCVolume.File
- Device.Process.File
- Device.File
- Device.IOSize
- Device.Process

The VSI PERFDAT OpenVMS data collector scans all active data collections and checks in the order listed above, if such metrics are enabled. If this is the case, it disables the first metric found for each collection and continues processing. At the end of the next sample interval it checks its CPU consumption again. If it is still above the threshold it disables the next metric in the list. It continues as long as its CPU consumption is above the CPU load threshold and as long as there is still one of the metrics listed above enabled for one of the active collections. If all these metrics have been disabled for all collections running and the CPU consumption is still above the defined threshold, the VSI PERFDAT OpenVMS data collector then stops the collection with the shortest sample interval. The data collector proceeds and stops every collection in the order of increasing sample intervals until the CPU consumption is below the threshold or no more collections are active.

The default of the CPU load threshold is 20% of the overall CPU power. The threshold can be modified by defining the system-wide logical PERFDAT$MAX_CPU_LOAD.

To set the threshold to 5% enter

$ DEFINE/SYSTEM PERFDAT$MAX_CPU_LOAD 5

I/O load triggered dynamic resource trimming

At the end of the sample interval of each collection the VSI PERFDAT OpenVMS data collector checks if all asynchronous database store operations (file writes) for the collection have been completed. If there are still some uncommitted transactions no data are sampled but the sample interval is doubled for the associated collection to give room to complete the database transactions. This is done a maximum of 3 times. If the collection interval for a specific collection interval has been doubled 3 times and there are still uncommitted inserts the collection will be stopped.

Under normal conditions it is very unlikely that either the CPU-based or the I/O based resource trimming will be triggered.

You can suppress the I/O load triggered dynamic resource trimming algorithm by assigning the value ‘TRUE’ to the system-wide logical PERFDAT$DO_NOT_INCREASE_SAMPLETIME

$ DEFINE/SYSTEM PERFDAT$DO_NOT_INCREASE_SAMPLETIME TRUE

Note

If you assign the value ‘TRUE’ to this logical be aware that in this case the OpenVMS data collector will not stop automatically in case of heavy I/O load although it might cause overall I/O performance problems.

VSI PERFDAT EVA extension

The VSI PERFDAT EVA extension collects performance data from HP StorageWorks Enterprise Virtual Arrays (EVA) according to the collection profiles the data collections were started with, and stores the data in the VSI PERFDAT performance database.

Features

- Up to 64 EVA arrays can be monitored in parallel
- Easy to configure
- Friendly Name Resolution
  The VSI PERFDAT EVA extension automatically resolves all monitored items (virtual disk, physical disk, host connections etc.) to their names assigned by CV/EVA (CommandView/EVA). The friendly name resolution of the EVA extension does not depend on the availability of the SAN-appliance or on any other external source. The friendly names are fetched directly from the accessed EVA(s).
- Automatic configuration change detection
  Whenever the configuration of the EVA changes (add/modify/ delete virtual disks, host connections etc.) the friendly name reference table is automatically reloaded. Thus, no management actions are required after the EVA configuration has changed – a manual friendly name reference table reload or a manual restart of the data collection is NOT required.
- Profile controlled
  Profiles reside, as with the other collection profiles for other systems (OpenVMS, Brocade switches etc.), in the VSI PERFDAT configuration database and are managed using the PERFDAT_MGR utility.
- Sample interval is freely definable (minimum = 5 sec.)
- Each metric can be enabled/disabled independently
- Fully integrated into the VSI PERFDAT environment
  All the advanced functions and features of VSI PERFDAT are available for data collected by the VSI PERFDAT EVA extension:
  - Automatic trend reports
  - Data management and housekeeping
  - Online alerting
  - All statistical methods (sorting, correlation and deviation analysis etc.) provided by the DQL interface can be applied to data collected by the EVA extension
  - Data can be accessed and analysed by using the DQL$ utility or the GUI as with any other data collected by the PERFDAT OpenVMS data collector or the PERFDAT SNMP extension.

Supported HP StorageWorks Enterprise Arrays and HSV controllers

The VSI PERFDAT EVA extension supports all currently available HP StorageWorks Enterprise Arrays and HSV controllers as listed below:

Controller Type	EVA Model	Minimum VCS/XCS version
HSV 100 HSV 110	EVA 3000 EVA 5000	VCS V3.028
HSV 200	EVA 4000 EVA 6000	XCS V5.030
HSV 200-B	EVA 4100 EVA 6100	Any XCS version
HSV 210	EVA 8000	XCS V5.030
HSV 300	EVA 4400	XCS 09522
HSV 340	EVA 6300/6350	Any XCS version
HSV 360	EVA 6500/6550	Any XCS version
HSV 400	EVA 6400	XCS 09522
HSV 450	EVA 8400	XCS 09522

Available metrics

The VSI PERFDAT EVA extension provides the metrics listed below:

Metric	Description
ARRAY	This metric provides system summary performance statistics of the EVA system
CTRL	This metric provides controller performance statistics for both HSV controllers of an EVA system
CTRL.PORT	This metric provides port specific performance statistics for each controller port of the EVA system
CTRL.HOSTCONN	This metric provide host connection performance statistics for each HSV controller of the EVA system. Host connections are automatically resolved to their friendly names without any preceding user action. This metric is not available for HSV 110 controllers running VCS 3.xxx.
DISKGROUP	This metric provides disk group performance statistics.

DISKGROUP.PDISK	This metric provides physical disk performance statistics grouped by the disk group each disk is a member of. Any physical disk is automatically mapped to its friendly name assigned by CV/EVA (CommandView/EVA). Performance data for ungrouped disks are not collected.
	Note This metric is not available for HSV 340 and HSV 360 controllers (EVA 6300/6350/6500/6550)
DISKGROUP.VDISK	This metric provides virtual disk performance statistics grouped by the disk group the virtual disk is a member of. Any virtual disk is automatically mapped to its friendly name assigned by CV/EVA (CommandView/EVA). Performance statistics are only collected for presented virtual disks. Not presented virtual disks are ignored.
DRM.TUNNEL	Data replication tunnel statistics.

General description

With the VSI PERFDAT EVA extension, up to 64 EVA systems can be monitored by one VSI PERFDAT OpenVMS system. EVA performance data are collected by up to 64 VSI PERFDAT EVA working processes named PERFDAT_EVA_x (where x = 0 …63). These working processes are managed by the VSI PERFDAT EVA master process named PERFDAT_EVA, which is controlled by the PERFDAT_MGR utility.

The VSI PERFDAT EVA master process keeps the status information of each working process and their respective active collections. The main task of the VSI PERFDAT EVA master process is to create the working processes, to dispatch *start*, *stop*, *show* and *shutdown* requests to the appropriate VSI PERFDAT EVA working processes.

PERFDAT_EVA_MASTER.EXE is the master process image and is located in the PERFDAT$BIN directory.

PERFDAT_EVA_WRK.exe is the working process image and is also located in the PERFDAT$BIN directory.

All run-time informational, warning and error messages are posted to OPCOM and are stored in log files. The log files of all the VSI PERFDAT EVA extension processes are located in the PERFDAT$LOG directory. The filename of the VSI PERFDAT EVA master process log-file has the format:

PERFDAT_EVA_*nodename*.LOG

The filename of a VSI PERFDAT EVA working process log file has the format:

PERFDAT_EVA__x_*nodename*.LOG (where x = 0 … 63)

where *nodename* defines the node where the VSI PERFDAT EVA extension is running on.

Startup/Shutdown

The VSI PERFDAT EVA extension can be started by any privileged user either via the PERFDAT_MGR utility (see chapterPERFDAT_MGR Management Interface):

```
$ MCR PERFDAT_MGR LAUNCH PERFDAT_EVA
```

or directly from the DCL command line by executing the interactive startup script:

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

The privileges required to start the VSI PERFDAT EVA extension are:

- CMKRNL
- NETMBX
- OPER
- SYSLCK
- SYSPRV
- TMPMBX
- WORLD

Note

Launching the VSI PERFDAT EVA extension with either commands listed above also starts the DQL interface (DQL$SRV, PDBC$SRV), the performance database filename cache service DQL_NAME and the auto archiving process.

VSI PERFDAT provides full multi-version support. This means that no system manager action is required to upgrade/install VSI PERFDAT if OpenVMS is upgraded to any version supported by VSI PERFDAT. The VSI PERFDAT EVA extension startup command script checks the actual OpenVMS version in use before starting the VSI PERFDAT EVA extension. If OpenVMS has been upgraded all VSI PERFDAT version specific images are replaced and loaded automatically

before the VSI PERFDAT EVA extension is started to avoid version mismatch problems.

In order to shutdown the VSI PERFDAT EVA extension, enter:

```
$ MCR PERFDAT_MGR SHUTDOWN PERFDAT_EVA
```

The VSI PERFDAT EVA extension will be shutdown implicitly if you shutdown the whole VSI PERFDAT environment with the command:

```
$ MCR PERFDAT_MGR SHUTDOWN ALL
```

For additional information about the actions performed when executing either of these commands, please refer to PERFDAT_MGR – Management Interface.

Initialization phase

During the initialization phase the VSI PERFDAT EVA master process performs the following actions:

- Creates and initializes mailboxes for online communication with the management interface PERFDAT_MGR
- Searches the configuration database for auto-start collections configured to run on the local node.
- Creates as many working processes as needed to start all the configured auto-start collections.
- Creates and initializes mailboxes for online communication with the working processes
- Dispatches the collections defined in the auto-start table of the VSI PERFDAT configuration database to the working processes.
- Checks if online performance alerting is enabled for any of the auto-start collections. If this is the case the working processes that execute such a data collection are triggered to invoke the online performance alerting subsystem.

Two mailboxes are created and initialized for online communication with the PERFDAT_MGR management utility as shown in Fig. 2.1. The master process receives management command via the PERFDAT_EVA_MST_IN mailbox and transmits status information via the PERFDAT_EVA_MST_OUT mailbox.

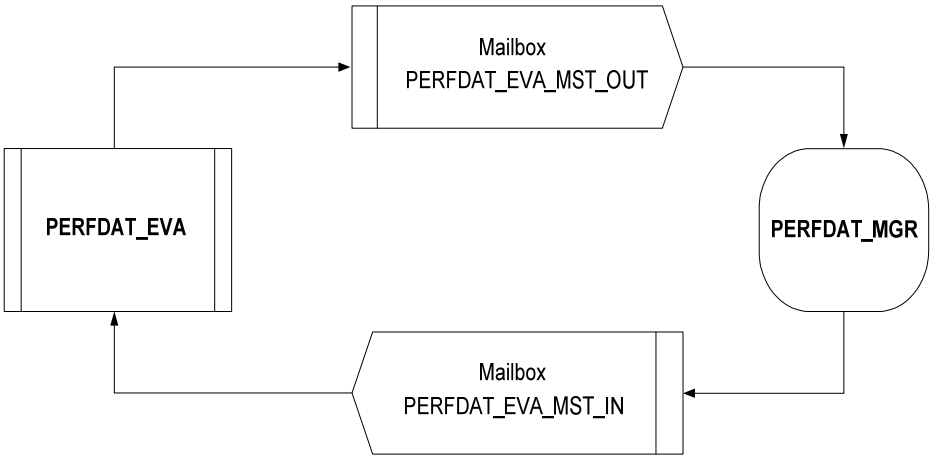


Fig. 2.1 Mailbox communication between PERFDAT_MGR and the master process of the VSI PERFDAT EVA extension

Mailboxes are also used for online communication between the master process and the working processes as shown in Fig. 2.2.

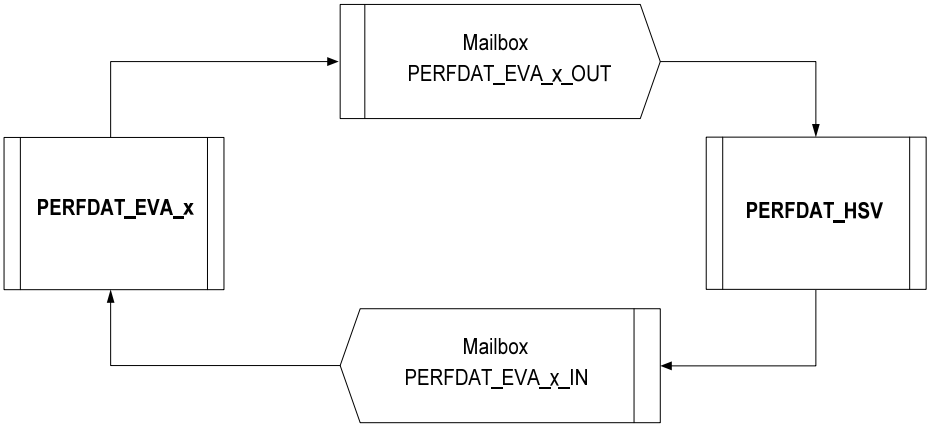


Fig. 2.2 Mailbox communication between the master process and the working processes of the VSI PERFDAT EVA extension (where x= 0 … 63)

EVA configuration

- Check if the EVA system to monitor is password protected. The VSI PERFDAT EVA extension cannot collect performance data from password protected EVA systems.
- The OpenVMS system running the VSI PERFDAT EVA extension has to have at least one adapter that is a member of the same Fibre Channel zone that the EVA system(s) is a part of.
- Login to CV/EVA (CommandView/EVA) and select the EVA system to be monitored. In to grant OpenVMS access to the console of the EVA, the 'Console LUN ID' parameter has to be greater than zero (zero is the default). Change the value of the parameter if it is zero.
- Make sure that the numbers assigned to the 'Console LUN ID' parameter of the EVA systems are unique.

- Run $ MCR SYSMAN IO AUTOCONFIGURE on the OpenVMS system. After you have executed the command, the 1GGAxxx devices will be available on the OpenVMS system for all unique Console LUN IDs (i.e. if you have assigned 90 to the Console LUN ID parameter of an EVA system you will get a device named 1GGA90 on OpenVMS). These devices will be used by the VSI PERFDAT EVA extension to access the controllers of the EVA system.
- Use the SHOW DEVICE 1GGAxxx/FULL DCL command to check if there is at least one path available to each HSV controller of the EVA systems. It does not matter if you have more paths available per HSV controller.
- Now you can configure your EVA performance data collections.

Starting a collection

Data collections are like the VSI PERFDAT OpenVMS data collector, profile controlled. A collection profile defines the sample interval and the metrics to collect. Collection profiles are stored in the VSI PERFDAT configuration database and are managed via the PERFDAT_MGR utility. The configuration database is:

PERFDAT$CFG:PERFDAT_PROFILES.CFG.

Data collections will be started either manually via the PERFDAT_MGR utility when the EVA extension is up and running or automatically after start-up (see chapter Initialization phase).

Whenever a collection is triggered via the PERFDAT_MGR interface the whole collection profile content is sent via Mailbox to the master process (see also VSI PERFDAT_MGR – Management Interface). The master process verifies if a performance collection is already in progress for the requested EVA system. If this is the case the master process rejects the start request. Otherwise it starts a working processes and forwards the start request to that working process..

The working process creates a new data collection file, accesses the record descriptor table of the VSI PERFDAT configuration database, reads the field descriptors for the enabled metrics and stores these descriptors in the header of the newly created file. The reason for copying the field descriptors into the data file is to guarantee read access to the data independent of:

- Version of the DQL environment
- Version of the VSI PERFDAT EVA extension
- A valid VSI PERFDAT configuration database exists on the access server
- The field descriptors stored in the VSI PERFDAT configuration database on the access server matches the data records

After initializing internal data structures the working process requests configuration data (MLD scan) from the EVA system. During that phase the collection state is “INITIAL CONFIG”. After the EVA system configuration data scan has succeeded and the working process starts collecting performance data the collection state changes to “ACTIVE”. The first sample is taken at the next full minute after the time the EVA system configuration scan has completed.

The actual states of the collections can be monitored with the PERFDAT_MGR SHOW COLLECTION command. For more details please refer to VSI PERFDAT_MGR – Management Interface or the VSI PERFDAT -PERFDAT_MGR Reference Manual.

Online performance alerting

Online performance alerting is performed by the online alerting subsystem and can be enabled for any active performance data collection. The online alerting subsystem is invoked by the EVA extension either on user request via the PERFDAT_MGR utility when the EVA master and working processes are up and running or automatically after start-up (see chapter Initialization phaseInitialization phase).

Data collection file

The working processes of the VSI PERFDAT EVA extension periodically create new data files for each active collection. The rules for creating new data files are:

- Whenever a collection is started (restarted)
- Data files are closed and new data files are created daily for each active collection. The time of day the collection data files are closed is defined when a collection is started.

Thus, 1 to n data file can exist per day and collection. The working processes of the VSI PERFDAT EVA extension create all data files in the directory referred to by the PERFDAT$DB_LOCAL logical. The file name format is:

PERFDAT_EVA_*node*_*yyyy-mm-dd*_*profile*.DAT_*sssss*

where:

- *node*
  Node name of the remote system
- *yyyy*
  Year the collection data file was created
- *mm*
  Month the collection data file was created
- *dd*
  Day the collection data file was created
- *profile*
  Collection profile used to start the performance data collection
- *sssss*
  Time since midnight when the collection data file was created

Any data collection file is created with the same default file size of approximately 80 MB. The file extent size is approximately 20MB. The system-wide logicals:

- PERFDAT$DATA_MEANSIZE

- PERFDAT$DATA_RECORDCNT

do not affect the initial file size (compare VSI PERFDAT OpenVMS Data Collector – Data collection file).

VSI PERFDAT SNMP extension

The VSI PERFDAT SNMP extension collects performance raw-data from non-OpenVMS systems that provide performance data via SNMP, converts this raw-data into human readable form, and stores the data in the performance data base.

Features

- Up to 64 remote nodes can be monitored in parallel
- Metrics and statistics are predefined for Tru64 systems and Brocade switches.
- Profile controlled – profiles reside in the VSI PERFDAT configuration database and are managed via the PERFDAT_MGR utility
- Sample interval is freely definable (minimum = 1 minute)
- Each metric can be enabled/disabled independently
- Permits online monitoring
- Online performance alerting can be enabled dynamically
- Controlled by PERFDAT_MGR

Available metrics

The VSI PERFDAT SNMP extension is designed to collect performance data of any remote node via SNMP. Collecting performance data via SNMP requires knowledge of the remote system's MIB table. In order to monitor a specific type of remote system, system specific configuration tables based on its MIB table have to be created and loaded into the VSI PERFDAT configuration database.

Currently such configuration tables are available for Tru64 systems, Solarissystems, Linux systems and Brocade switches. The creation of configuration tables for any other system (such as network components, other SAN switches etc.) is not covered by this documentation.

Below all metrics available for Tru64 are listed. A detailed description of the statistics captured by each metric is provided in Appendix A of this document

- Tru64_System
- Tru64_CPU
- Tru64_Process
- Tru64_Deamon
- Tru64_User
- Tru64_Disk
- Tru64_FileSys
- Tru64_NIC
- Tru64_IP

Below all metrics available for Brocade switches are listed. A detailed description of the statistics captured by each metric is provided in Appendix A of this document

- Port
- System
- System.Fan
- System.Temperature

Below all metrics available for Solaris systems are listed. A detailed description of the statistics captured by each metric is provided in Appendix A of this document

- Sun_Device
- Sun_Process
- Sun_Deamon
- Sun_NIC
- Sun_IP
- Sun_TCP
- Sun_FileSys
- Sun_System

Below all metrics available for Brocade switches are listed. A detailed description of the statistics captured by each metric is provided in Appendix A of this document

- Linux_Process
- Linux_Deamon
- Linux_NIC
- Linux_IP
- Linux_TCP
- Linux_FileSys
- Linux_System

General description

With the VSI PERFDAT SNMP extension, up to 64 remote nodes providing performance data via SNMP, can be monitored. Performance data is collected by up to 8 PERFDAT SNMP working processes named PERFDAT_SNMP_x (where x = 0 … 7). Each working process can handle up to 8 data collections. These working processes are controlled by a VSI PERFDAT SNMP master process named PERFDAT_SNMP, which is controlled by the PERFDAT_MGR management utility.

The VSI PERFDAT SNMP master process keeps the status information of each working process and their respective active collections. The main task of the VSI PERFDAT SNMP master process is to dispatch start, stop and show collection request as well as shutdown requests to the appropriate working process.

PERFDAT_SNMP_MASTER.EXE is the master process image and is located in the PERFDAT$BIN directory.

PERFDAT_SNMP_WRK.EXE is the working processes image and is also located in the PERFDAT$BIN directory.

All informational, warning and error messages produced during runtime are posted to OPCOM and stored in log files. The log files of all processes of the VSI PERFDAT SNMP extension are located in the PERFDAT$LOG directory. The filename of the master process log-file has the format:

PERFDAT_SNMP_*nodename*.LOG

The filename of the working processes log file has the format:

PERFDAT_SNMP__x_*nodename*.LOG (where x = 0 … 7)

where *nodename* defines the node the VSI PERFDAT SNMP extension is running.

Startup / Shutdown

The VSI PERFDAT SNMP extension can be started by any privileged user either via the PERFDAT_MGR utility (see chapterPERFDAT_MGR Management Interface):

```
$ MCR PERFDAT_MGR LAUNCH PERFDAT_SNMP
```

or directly from the DCL command line by executing the interactive startup script:

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

The privileges required to start the VSI PERFDAT SNMP extension are:

- CMKRNL
- NETMBX
- OPER
- SYSLCK
- SYSPRV
- TMPMBX
- WORLD

Note

Launching the VSI PERFDAT SNMP extension with either commands listed above also starts the DQL interface (DQL$SRV, PDBC$SRV), the performance database filename cache service DQL_NAME and the auto archiving process.

VSI PERFDAT provides full multi-version support. This means that no system manager action is required to upgrade/install VSI PERFDAT if OpenVMS is

upgraded to any version supported by VSI PERFDAT. The VSI PERFDAT SNMP extension startup command script checks the actual OpenVMS version in use before starting the VSI PERFDAT SNMP extension. If OpenVMS has been upgraded all VSI PERFDAT version specific images are replaced and loaded automatically before the VSI PERFDAT SNMP extension is started to avoid version mismatch problems.

Fig. 2.3 shows the start-up flow-chart for the VSI PERFDAT SNMP extension. The start-up sequences are identical if the user invokes either of the command scripts or if the PERFDAT_MGR management utility is used to launch the VSI PERFDAT SNMP extension.

In order to shutdown the VSI PERFDAT SNMP extension, enter:

```
$ MCR PERFDAT_MGR SHUTDOWN PERFDAT_SNMP
```

The VSI PERFDAT SNMP extension will be shutdown implicitly if you shutdown the whole VSI PERFDAT environment with the command:

```
$ MCR PERFDAT_MGR SHUTDOWN ALL
```

For additional information about the actions performed when executing either of these commands, please refer to chapter VSI PERFDAT_MGR – Management Interface.

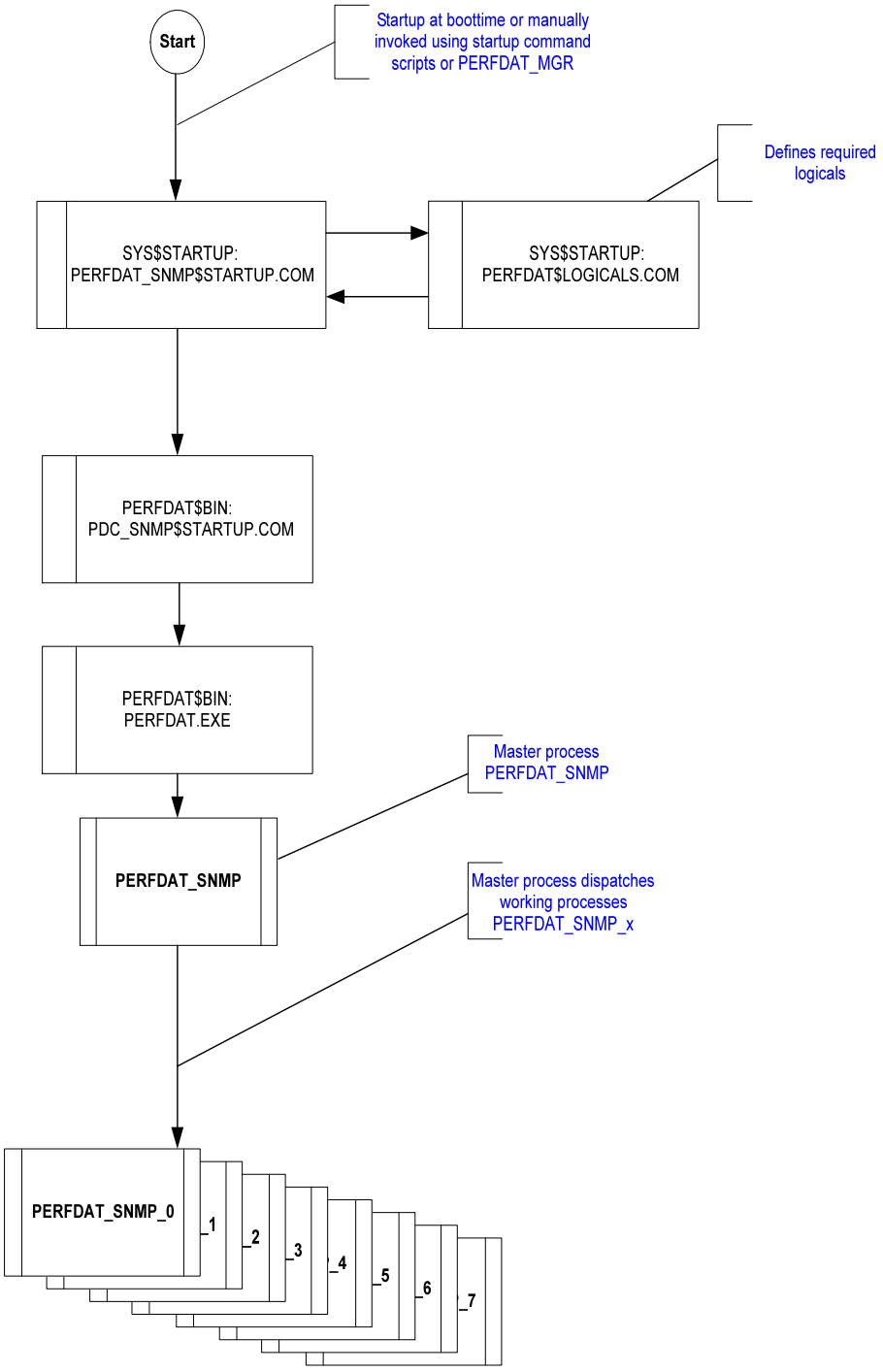


Fig. 2.3 Start-up flow-chart for the VSI PERFDAT SNMP extension.

Initialization phase

As you can see from Fig. 2.3 the start-up command scripts creates the master process PERFDAT_SNMP. During the initialization phase the master process performs the following actions:

- Creates and initializes mailboxes for online communication with the management interface PERFDAT_MGR

- Initializes the logical PERFDAT$SNMP_BASE_PORT
  This logical defines the UDP port number to be use by the working process when starting a new collection. If the logical is not defined it is set to 2500. Otherwise it is left unchanged.
- Searches the configuration database for auto-start collections configured (= remote systems to monitor and the collection profile to use) to run on the local node.
- Creates as many working processes as needed to start all configured auto-start collections.
- Creates and initializes mailboxes for online communication with the working processes
- Dispatches the collections to the working processes.
- Checks if online performance alerting is enabled for any of these auto-start collections. If this is the case the working processes that executes such a data collection is triggered to invoke the online performance alerting subsystem.

Two mailboxes are created and initialized for online communication with the PERFDAT_MGR management utility as shown in Fig. 2.4. The master process receives management command via the PERFDAT_SNMP_MST_IN mailbox and transmits status information via the PERFDAT_SNMP_MST_OUT mailbox.

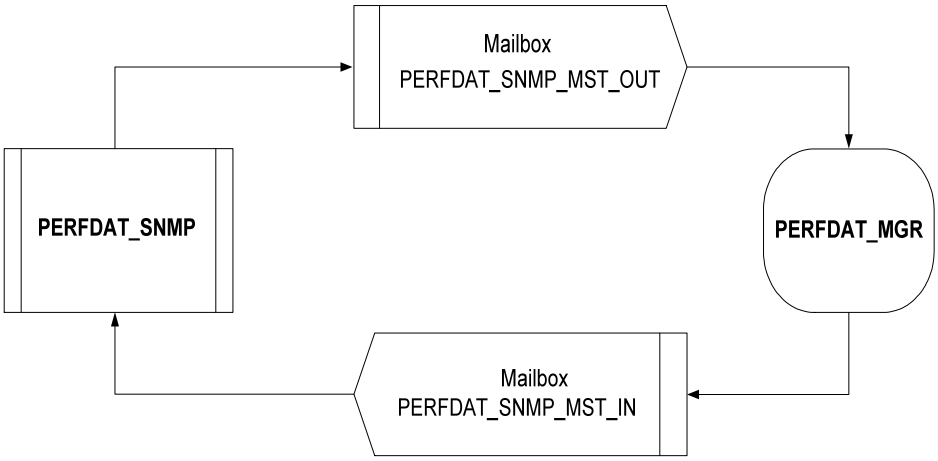


Fig. 2.4 Mailbox communication between PERFDAT_MGR and the master process of the VSI PERFDAT SNMP extension

Mailboxes are also used for online communication between the master process and the working processes as shown in Fig. 2.5.

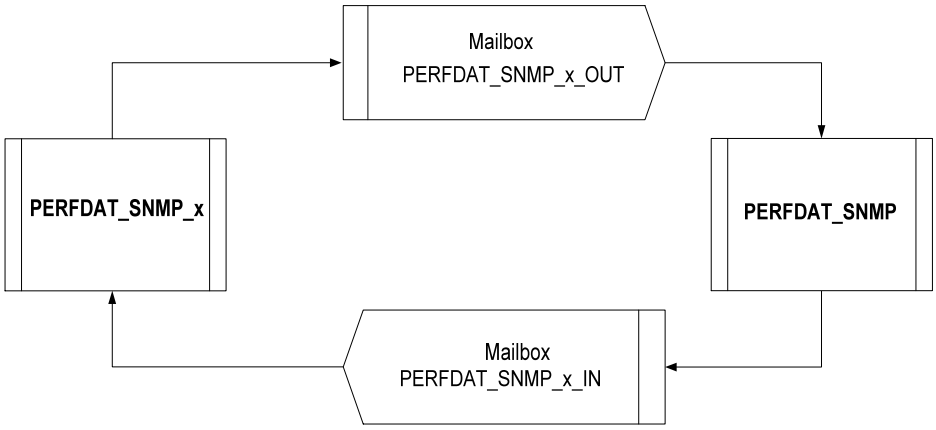


Fig. 2.5 Mailbox communication between the master process and the working processes of the VSI PERFDAT SNMP extension (where x= 0 … 7)

Brocade switch configuration

- Make sure that the IP address of the SNMP agent that is used to run the Brocade switch performance data collection is added to the SNMP access list of the target Brocade switch
- Check the SNMP community names configured on the target Brocade switch. Use one of these community names to start the Brocade switch performance data collection.

For detailed information about how to define the community string and the source IP address when starting a SNMP performance data collection please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference manual.

Brocade switch access test

The test utility

PERFDAT$TOOLS:BROCADE_TEST.COM

provided by VSI PERFDAT can be applied to test whether or not a Brocade switch provides valid responses for all OID requests required to run a SNMP performance data collections.

Run this utility before you start any SNMP data collection for your target Brocade switch.

This test utility requires two input parameter:

- P1 IP address or node name of the target system
- P2 Community name
  Enter the SNMPv1/SNMPv2 community name. It is strongly recommended to enter the community name with quotation marks since the SNMP community string check is case sensitive. If you omit quotation marks the community name entered is converted to upper case.

If P2 is not applied, "FibreChannel" will be used as the default community name for the test.

Example:

To start the Brocade switch access test for the FC switch FCSW201 with community name “public”:

@PERFDAT$TOOLS:BROCADE_TEST FCSW201 “public”

Tru64 configuration

- Make sure that SNMP is started on the target system and that the PMGRD daemon is up and running.
- Check the SNMP communities configured on the target system. Use one of the available read only SNMP communities to start the SNMP performance data collection.

For detailed information about to configure and start SNMP and the PMGRD daemon on Tru64 please refer to the appropriate documentation of Tru64.

For detailed information about how to define the community string when starting a SNMP performance data collection please refer to the manual VSI PERFDAT – PERFDAT_MGR Reference manual.

Solaris configuration

The prerequisite to run performance data collections via the VSI PERFDAT SNMP extension for Solaris systems is that NET-SNMP is installed and running on the remote Solaris system. NET-SNMP is an open-source SNMP daemon that is available for many UNIX systems. The NET-SNMP daemon provides a wide range of system performance statistics.

If you are running Solaris 10 the NET-SNMP does not have to be installed since NET-SNMP is part of the distribution kit. If you are running an older Solaris version (2.6, 7, 8 or 9) you have to install NET-SNMP on the target system before you can run performance data collections via the VSI PERFDAT SNMP extension.

Before you install NET-SNMP V5.1.2 please read the documentation (also available from the download Web site)

PERFDAT$SUPPORT:README_solaris_binaries_512.TXT

Add a SNMPv1/SNMPv2 read-only community name to the access control setup section in the *snmpd.conf* file (on the Solaris system) that will be used by the VSI PERFDAT SNMP extension to collect performance data from the target Solaris system. You can add any community name. VSI PERFDAT SNMP data collection

has to be configured to use the community name you have entered in the *snmpd.conf* file on the Solaris system.

Example:

To add the SNMPv1/SNMPv2c read-only access community name “perfdat” add the line

```
rocommunity perfdat
```

to the access control section of the *snmpd.conf* file as shown below.

```
###########################################################
# SECTION: Access Control Setup
#
# This section defines who is allowed to talk to your running
# snmp agent.

# rocommunity: a SNMPv1/SNMPv2c read-only access community name
# arguments: community [default|hostname|network/bits] [oid]

rocommunity perfdat
```

The *snmpd.conf* file is either located in */etc/snmp/conf/* (Solaris 10) or in */usr/local/share/snmp/* (Solaris 2.6, 7, 8, 9 with NET-SNMP V5.1.2 installed). If the *snmp.conf* does not exist, create it add the access control section as shown above.

Finally start the NET-SNMP daemon.

For additional information and documentation about NET-SNMP please refer to http://www.net-snmp.org.

Linux configuration

The prerequisite to run performance data collections via the VSI PERFDAT SNMP extension for Linux systems is that NET-SNMP is installed and running on the remote Linux system. NET-SNMP is an open-source SNMP daemon that is available for many UNIX systems. The NET-SNMP daemon provides a wide range of system performance statistics.

Certain Solaris and Linux distributions kits such as

- RedHat Enterprise Linux 4
- RedHat Enterprise Linux 5

contain a NET-SNMP package, which will be automatically installed and started if you install and configure SNMP on these systems.

Not all Linux distributions have been checked to see whether or not the NET-SNMP daemon is integrated as part of the distribution. As stated previously

RedHat Enterprise Linux 4 & 5 contains NET-SNMP as the default SNMP daemon.

If you run any other Linux distribution make sure that NET-SNMP is installed. If NET-SNMP is not installed, download the most recent kit from:

http://www.net-snmp.org

and install it on your target Linux system.

Add a SNMPv1/SNMPv2 read-only community name to the access control setup section in the *snmpd.conf* file (on your Linux system) as described in the previous section and start the NET-SNMP daemon.

The *snmpd.conf* file is either located in:

- */usr/local/share/snmp/*
- */etc/snmp/conf/*
- */etc/snmp/*

depending whether or not NET-SNMP is an integrated part of the Linux distribution.

Solaris and Linux configuration test

The test utility

PERFDAT$TOOLS:NET-SNMP_TEST.COM

provided by VSI PERFDAT can be applied to test whether or not the NET-SNMP daemon installed on the target system provides valid responses for all OID requests required to run a SNMP performance data collections for Solaris or Linux.

Run this utility before you start any SNMP data collection for your target Solaris or Linux system.

This test utility requires three input parameter:

- P1 target system type
  Valid keywords for P1 are SOLARIS or LINUX. If you want to test the NET-SNMP installation on a Solaris system enter SOLARIS, if you want to test the NET-SNMP installation on a Linux system enter LINUX.
- P2 IP address or node name of the target system
- P3 Community name
  Enter the SNMPv1/SNMPv2 community name you have defined in the *snmpd.conf* file on the target system. It is strongly recommended to enter the community name with quotation marks since the SNMP community string check is case sensitive. If you omit quotation marks the community name entered is converted to upper case.

Example:

To start the NET-SNMP daemon response test for the Linux system PLUTO with community name “perfdat”:

@PERFDAT$TOOLS:NET-SNMP_TEST LINUX PLUTO “perfdat”

Starting a collection

Data collections are like the VSI PERFDAT OpenVMS data collector, profile controlled. A collection profile defines the sample interval and the metrics to collect. Collection profiles are stored in the VSI PERFDAT configuration database and are managed via the PERFDAT_MGR utility. The configuration database is:

PERFDAT$CFG:PERFDAT_PROFILES.CFG.

Note

With the VSI PERFDAT SNMP extension only one collection can be active per remote node. In other words, if any working process runs a performance data collection for a dedicated remote system, you cannot start another collection – independent of the fact if the collection profile differs from the active one or not.

Data collections will be started either manually via the PERFDAT_MGR utility when the SNMP extension is up and running or automatically after start-up (see chapter Initialization phase).

Whenever a collection is triggered via the PERFDAT_MGR interface the whole collection profile content is sent via Mailbox to the master process (see also VSI PERFDAT_MGR – Management Interface). The master process verifies if a performance collection is already in progress for the requested node. If this is the case the master process rejects the start request. Otherwise it queries the working processes for free collection slots (please note each working process can collect performance data for up to 8 remote systems), and forwards the start request to that working process that signaled a free collection slot. If more than one working process has signaled free collection slots the master process forwards the start request to that working process with the lowest working ID (the working ID is the number visible at the end of the working process name). E.g. if working process PERFDAT_SNMP_1 and PERFDAT_SNMP_3 signaled free collection slots, the master process forwards the request to process PERFDAT_SNMP_1. Afterwards the master process checks if there are still free collection slots. If there are no free slots, it creates a new working process if the limit of 8 working processes has not been reached.

Once a working process has received a start request it first creates a new socket with the port number defined by the system-wide logical PERFDAT$SNMP_BASE_PORT.

Note

Be careful when redefining the PERFDAT$SNMP_BASE_PORT logical. Ensure that the port number assigned to the logical is not in use. Otherwise the start of a new collection will fail.

The working process then creates a new data collection file, accesses the record descriptor table of the VSI PERFDAT configuration database, reads the field descriptors for the enabled metrics and stores these descriptors in the header of the newly created file. The reason for copying the field descriptors into the data file is to guarantee read access to the data independent of:

- Version of the DQL environment
- Version of the VSI PERFDAT SNMP extension
- A valid VSI PERFDAT configuration database exists on the access server
- The field descriptors stored in the VSI PERFDAT configuration database on the access server matches the data records

After initializing internal data structures the working process requests configuration data such as OS version or firmware revision from the remote node. It waits 15 seconds to receive an answer and sets the collection state to “INITIAL CONFIG”. If the working process gets no valid answer, it stalls the collection and periodically retriggers the configuration query (10 minutes interval) until the remote system provides a valid response or the collection is manually stopped. After receiving a valid response to the configuration query the collection state changes to “ACTIVE” and the working process starts collecting data from the remote system. The first sample is taken at the next full minute after the time the configuration response was received.

If the communication between the working process and a remote system breaks for any reason (e.g. system shutdown of the remote system, network link breaks etc,) during data collection, the collection will not be stopped. The collection state changes to “NOT RESPONDING”. The working process still tries to get performance data at every sample interval. The first time the remote system responds again the collection state changes back to “ACTIVE”. During the time the collection is in the “NOT RESPONDING” state no data is written.

The actual states of the collections can be monitored with the PERFDAT_MGR SHOW COLLECTION command. For more details please refer to VSI PERFDAT_MGR – Management Interface or the VSI PERFDAT -PERFDAT_MGR Reference Manual.

Online alerting

Online performance alerting is performed by the online alerting subsystem and can be enabled for any active performance data collection. The online alerting subsystem is invoked by the SNMP extension either on user request via the PERFDAT_MGR utility when the SNMP master and working processes are up and running or automatically after start-up (see chapter Initialization phaseInitialization phase).

Data collection file

The working processes of the VSI PERFDAT SNMP extension periodically create new data files for each active collection. The rules for creating new data files are:

- Whenever a collection is started (restarted)
- Data files are closed and new data files are created daily for each active collection. The time of day the collection data files are closed is defined when a collection is started.

Thus, 1 to n data file can exist per day and collection. The working processes of the VSI PERFDAT SNMP extension create all data files in the directory referred to by the PERFDAT$DB_LOCAL logical. The file name format is:

PERFDAT_SNMP_*node_yyyy-mm-dd_profile*.DAT_*sssss*

where:

- *node*
  Node name of the remote system
- *yyyy*
  Year the collection data file was created
- *mm*
  Month the collection data file was created
- *dd*
  Day the collection data file was created
- *profile*
  Collection profile used to start the performance data collection
- *sssss*
  Time since midnight when the collection data file was created

Any data collection file is created with the same default file size of approximately 9 MB. The file extent size is approximately 2MB. The system-wide logicals:

- PERFDAT$DATA_MEANSIZE
- PERFDAT$DATA_RECORDCNT

do not affect the initial file size (compare VSI PERFDAT OpenVMS Data Collector – Data collection file).

VSI PERFDAT programming interface

VSI PERFDAT provides an easy to use C programming interface (API) to insert any type of performance data collected by the components (programs) of an application directly into the distributed VSI PERFDAT performance database.

The VSI PERFDAT installation procedure provides two object libraries that contain the API routines:

- PERFDAT$LIBRARY:PERFDAT_API_AXP.OLB
  Alpha object library
- PERFDAT$LIBRARY:PERFDAT_API_IA64.OLB
  I64 object library

Features

The use of the VSI PERFDAT API provides several advantages:

- The programmer does not have to worry about when to open or close a data file. Data files are automatically created, opened and closed as defined by the VSI PERFDAT design rules.
- The VSI PERFDAT environment handles data files created by an application using the API as if these data files had been created by any of the VSI PERFDAT data collectors (OpenVMS, SNMP extension, EVA extension).
    - Application data files are automatically managed by the VSI PERFDAT archive and housekeeping process reliable and unattended (for more information about VSI PERFDAT archiving and housekeeping please refer to the manual *VSI PERFDAT- Architecture and Technical Description*)
    - Trend, capacity and baseline report profiles can be defined for application data collections. These reports are automatically processed by the VSI PERFDAT auto trend engine (for more information about the VSI PERFDAT auto trend engine please refer to the manual *VSI PERFDAT- Architecture and Technical Description*).
- The API does not create separate data files for each process of an application but inserts the data provided by all processes of an application running on the same node into the same data file. This feature reduces the number of data files.
- Application data collections can be managed with the VSI PERFDAT management utility PERFDAT_MGR in the same way using the same commands as if one was managing OpenVMS, SNMP or EVA data collections without any programming effort, code change or the need of restarting the application.
    - Application data collections can be stopped at any time.
    - Application data collections are profile controlled as with other data collection created by one of the VSI PERFDAT data collectors (OpenVMS data collector, SNMP extension, EVA extension). Application data collection profiles can be user defined. A collection profile defines the sample interval and the

metrics that will be enabled when a data collection is started using a particular collection profile.
    - Once an application data collection has been stopped it can be started again with a different collection profile.
    - Online alerting can be enabled or disabled during run-time.
    - The status of application data collections can be monitored.
- Time concurrency
  Performance data is typically provided as averaged values like MB/sec or Transactions/sec. If an application consists of several processes which provide performance data as averaged values, it is important in terms of performance analysis that all these processes gather, calculate and provide the data at the same time so that this data can be compared and correlated to each other without any preprocessing. The VSI PERFDAT API triggers all processes of an application at the same time to collect, to calculate and to insert the data records into the metrics of the collection database regardless on which node the processes are running on within an OpenVMS cluster. Thus, the program developer does not have to care about such timing issues as described herein.

Using the programming interface

It is not in the scope of this document to describe the usage of the VSI PERFDAT programming interface. For a detailed description of how to use the VSI PERFDAT programming interface and a detailed description of the interface routines please refer to the manual VSI PERFDAT – Application Programming Interface User's Guide.

VSI PERFDAT distributed performance database

Database Organization

All data collected by the VSI PERFDAT OpenVMS data collector, the VSI PERFDAT EVA extension and the VSI PERFDAT SNMP extension are stored in index sequential RMS files. As has been described in the previous sections each data collector creates a new file daily or whenever a collection is started (restarted). Thus, 1 to n data files can exist per day and collection. A single data file is called a physical storage area. All physical storage areas that are created on the same day and that belong to the same collection (collection profile & node) is called a logical storage area. All logical storage areas created by the same data collection make up a collection database. The sum of all collection databases available within your environment is called the VSI PERFDAT distributed performance database (see Fig.2.6).

The database alias for each collection database is automatically assigned and cannot be changed by the user. The format of the alias is:

Nodename_collection-profile

E.g. an OpenVMS performance collection using a collection profile DEFAULT running on node BCSXTC creates and accesses the collection database BCSXTC_DEFAULT.

Trend and capacity report data files contain data of a particular time period – called a report period (day, week, month, quarter, year – for more information see VSI PERFDAT configuration database and VSI PERFDAT Auto Trend Engine) - defined by the report profile used to create these reports. At the end of such a predefined time period a new report data file is created, or whenever the statistics defined in the report profile changes. A single report data file is called a physical storage area. A logical storage is the sum of all physical storage areas that are created during a report period. E.g. if the report period is WEEK, all report data files created during a week make up the logical storage area for this report.

The VSI PERFDAT OpenVMS data collector, the VSI PERFDAT EVA extension and the VSI PERFDAT SNMP extension write the data directly via RMS calls to the appropriate data files. The trend engine does not directly access the data file, but inserts data via the DQL interface. Thus, as long as data files are write accessed by the VSI PERFDAT OpenVMS data collector, the EVA extension and the VSI PERFDAT SNMP extension these files have to be stored locally. Data files, that are write accessed by the auto-trend engine or by the DQL$ utility can be stored on any member of the community (logical PERFDAT$COMMUNITY) the local node belongs to.

The VSI PERFDAT OpenVMS data collector as well as the VSI PERFDAT EVA extension and the VSI PERFDAT SNMP extension create the data files in the directory pointed to by the logical PERFDAT$DB_LOCAL on the local node.

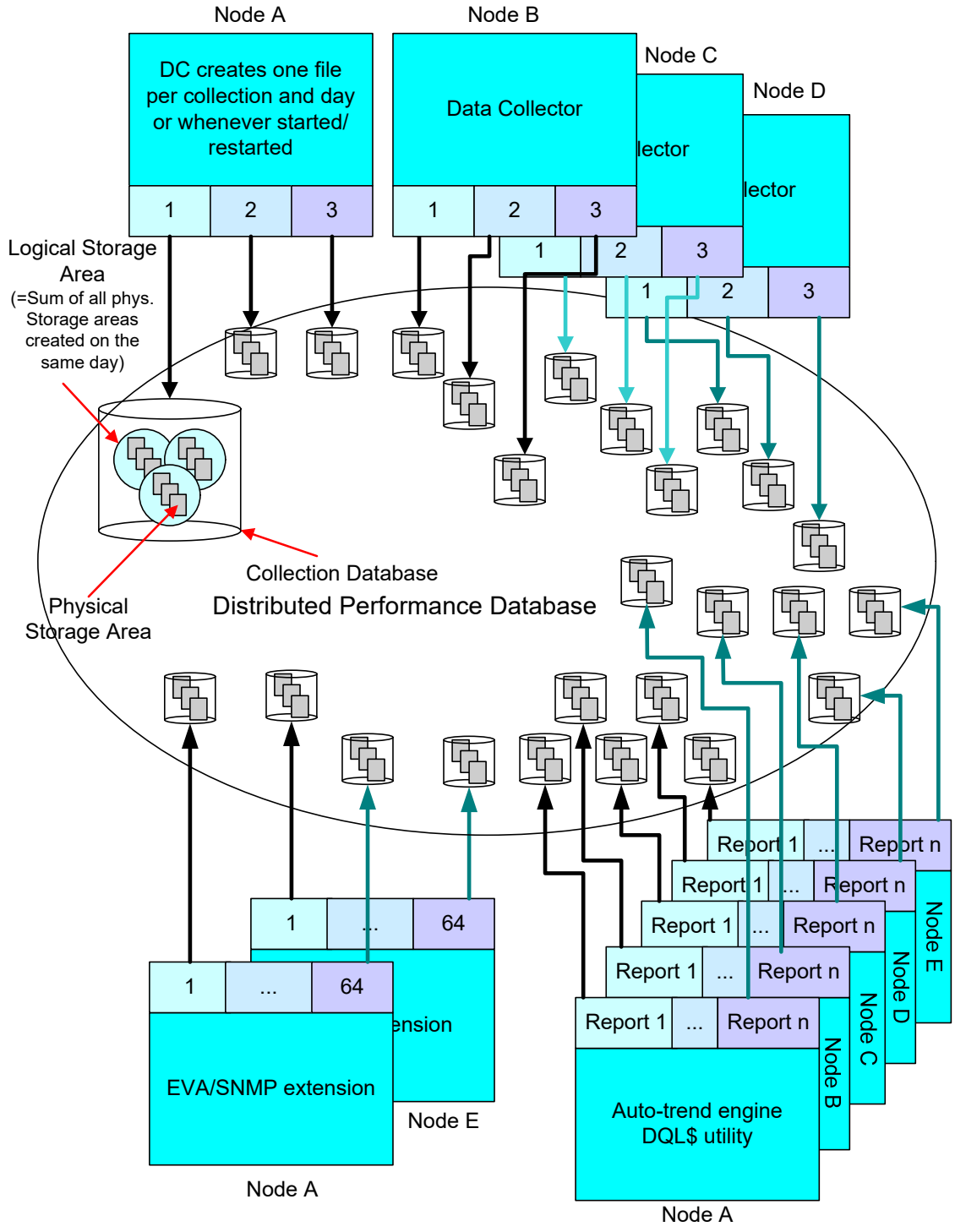


Fig. 2.6 Database organization of the VSI PERFDAT distributed database.

Report data files created by the auto-trend engine as well as the data files (physical storage areas) created by the DQL$ utility are stored initially on the local node too. However in contrast to the collection data files created by the VSI PERFDAT OpenVMS data collector and the VSI PERFDAT EVA/SNMP extension these files can be moved to another node within the PERFDAT community (e.g. archive node) after write access has been released by the auto-

trend engine or the DQL$ utility, even if these files are reopened later on for data insert, since the data is always accessed via the DQL interface.

Thus, report data files can be relocated between subsequent report-runs to another node of the PERFDAT community even if the auto-trend engine inserts data into the (relocated) data files again the next time.

The query interface to the distributed performance database is similar to SQL. All basic query statements such as SELECT, INSERT, CREATE, DROP etc. except UPDATE and DELETE to prevent after image data manipulation are supported (for more information regarding the command syntax please see chapter VSI PERFDAT Query Interface (DQL), VSI PERFDAT – DQL$ Reference manual or the DQL$ online help). The main difference between a relational database such as Oracle, Informix, mySQL etc. is that no root file exists for each database. All meta-data (field and record descriptors, data link descriptors, index reference table descriptor …) necessary to access the data is stored in the header of each physical storage area.

The advantage is that even if a physical storage area is moved to any other OpenVMS node (target node) that is not member of the community then that physical storage stays read accessible to DQL without any additional actions such as data conversion, unload and load operations. Thus, any data file can be copied to any OpenVMS node (target node) for offline analysis. The only prerequisite is that the DQL environment is installed on that OpenVMS node and it is configured as an archive node, or you simply define one of these logicals:

- PERFDAT$NODEDATA_HOSTED
- PERFDAT$NO_NODE_FILTER

For more information about these logicals see chapter VSI PERFDAT Directory structure and Logicals. OpenVMS and VSI PERFDAT version compatibility (= same versions) between the nodes data is collected and the node data is analyzed is NOT required.

Data within physical storage areas are organized in METRICES, ELEMENTS and STATISTICS. A METRIC consists of 1 to n ELEMENTS, and each ELEMENT consists of 1 to n STATISTICS.

Comparing this structure to a classic database organization the following comparisons are valid

- A METRIC is comparable to a TABLE.
- An ELEMENT is comparable to an INDEX of a TABLE.
- A STATISTIC is comparable to a FIELD within a TABLE.

Directory structure

Since no root file is involved for accessing the data files, directories are not freely definable. They have to be stored in one of the directories listed below.

- PERFDAT$DB_LOCAL

- PERFDAT$DB_ARCHIVE
- PERFDAT$DB_TREND
- PERFDAT$DB_SAVE
- PERFDAT$DB

These logical directories have to exist on any node within the VSI PERFDAT environment. Otherwise some or all VSI PERFDAT components may fail.

PERFDAT$DB_LOCAL

This is the default directory for creating performance collection data files. The VSI PERFDAT OpenVMS data collector, the VSI PERFDAT EVA extension and the VSI PERFDAT SNMP extension create files in this directory. If the logical does not exist or does not reference a valid physical directory, the data collectors fail.

PERFDAT$DB_ARCHIVE

All closed data files created by the VSI PERFDAT OpenVMS data collector, the VSI PERFDAT EVA extension and the VSI PERFDAT SNMP extension are periodically moved to this directory by the archiving process. The data files in this directory are managed by the archiving process. It guarantees that all files are kept for a predefined period of time (default 30 days – for more information see VSI PERFDAT Archiving and Housekeeping). Physical storage areas that are older than the defined keep time are automatically and unconditionally deleted. If this logical does not exist, or the logical does not refer a valid physical directory the archiving process fails.

PERFDAT$DB_TREND

Report data files are created in this directory. Between two subsequent report runs, the report data file can be moved to the same directory on any other node within the same community (e.g. archive node) and the report data file stay write accessible to the auto-trend engine as explained in the previous sections. If the logical does not exist or the logical does not refer to a valid physical directory the auto-trend engine as well any report extraction manually done via the DQL$ utility will fail.

PERFDAT$DB_SAVE

This directory is used for defining baseline performance data. A baseline is a set of logical storage areas (performance collection data) that covers a full week. The baseline represents a week where the system performance is deemed normal based on the customer's knowledge and experience. The baseline has to be defined by the system manager by moving the appropriate logical storage areas of a collection database to this directory. If the logical directory does not exist or the logical does not reference a valid physical directory a performance baseline cannot be defined. Thus, baseline deviation reports will fail.

PERFDAT$DB

When accessing the distributed performance database the DQL interface scans this directory on each member of the community for collection databases of the community members. PERFDAT$DB is a directory search list. Per default PERFDAT$DB refers to:

- PERFDAT$DB_LOCAL
- PERFDAT$DB_ARCHIVE
- PERFDAT$DB_TREND
- PERFDAT$DB_SAVE

This search list can be extended at any time. If this logical is not defined or it does not refer to valid logical or physical directories the whole database or parts of it will not be accessible via the DQL interface.

VSI PERFDAT configuration database

The VSI PERFDAT configuration database PERFDAT_PROFILES.CFG consists of 6 tables as shown in Fig. 2.7.

- Archive control table
- Autos-start table
- Collection profile table
- License table
- Record descriptor table
- Report profile table
- Stored procedure table
- Regional setting table

It is located in the directory pointed to by the PERFDAT$CFG logical name.

All tables of the VSI PERFDAT configuration database but the stored procedure table and the regional setting table are maintained by the PERFDAT_MGR management utility (for detailed information see chapter PERFDAT_MGR – Management Interface. The stored procedure table and the regional setting table are maintained by the DQL$ utility.

Archive control table

The archive control table stores initial control parameters for the archiving process PERFDAT_ARCHIVE of the VSI PERFDAT environment. This table is only accessed by the archiving process during start-up. None of the other VSI PERFDAT processes ever accesses this table.

Auto-start table

This table is accessed by the VSI PERFDAT OpenVMS data collector, the VSI PERFDAT EVA extension, the VSI PERFDAT SNMP extension and the VSI PERFDAT auto-trend engine (see Fig. 2.7).

The auto-start table of the VSI PERFDAT configuration database contains all required start-up parameters to start performance data collections for all nodes referred by its entries automatically when launching the OpenVMS data collector, the VSI PERFDAT EVA extension or the VSI PERFDAT SNMP extension. When the VSI PERFDAT environment is launched (re-launched) the OpenVMS data collector as well as the VSI PERFDAT SNMP extension and the VSI PERFDAT EVA extension accesses this table, checks if any performance data collections are defined to be started on the local node and starts them up automatically. This is done by checking the content of every entry in this table.

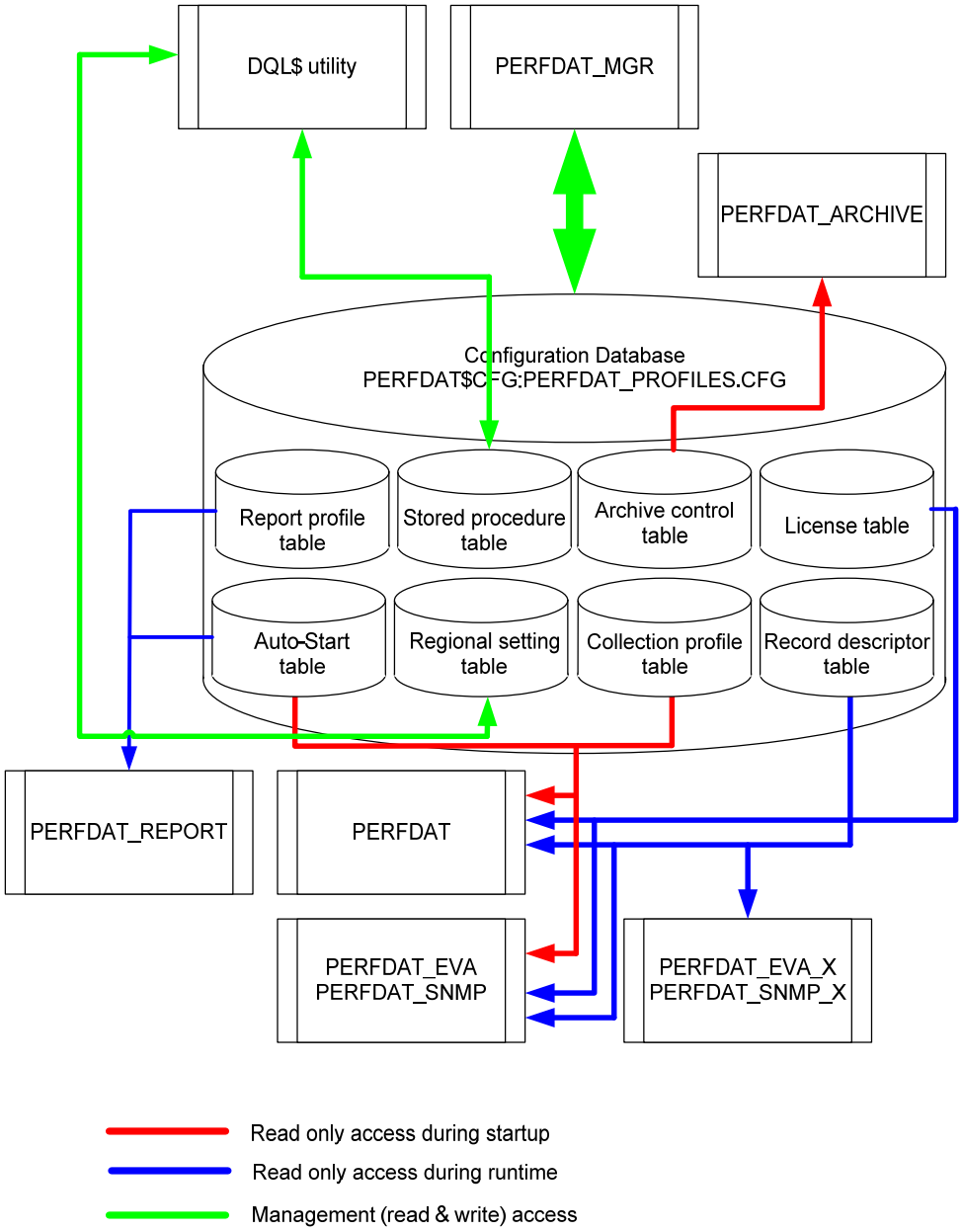


Fig. 2.7: Components of the VSI PERFDAT configuration database and process access to the tables of the database.

The entries of that auto-start table are also read by the auto-trend engine to determine if any trend and capacity report shall be processed. It checks if the local node itself is registered in the auto-start table and/or if the local node is defined as the SNMP agent (=node that runs a remote collection via SNMP) for any remote system (Tru64, Brocade) registered therein. If this is the case the reports defined in the report profile table of the VSI PERFDAT configuration database that are marked to be processed by the auto-trend engine and valid for these auto-start entries are processed (the auto-trend engine checks if the OS type of the auto-started performance collection matches the OS type of the report profiles).

A data record contains common and system specific fields. The common fields are:

- Collection profile name
  This field defines the collection profile to start automatically and refers the data collection files for trend and capacity report processing performed by the auto-trend engine.
- Access flag
  This field defines if the performance collection started automatically shall be online accessible or not. For all entries that refer to the VSI PERFDAT OpenVMS data collector and the VSI PERFDAT EVA extension the default value is FALSE, for all entries that refer to the VSI PERFDAT SNMP extension the default is TRUE.

Note

It is not recommended to allow online access to OpenVMS data collections. The reason is that depending on the collection profile settings huge amount of data will be written periodically (min 1 sec sample interval) to the data file. Since data files have to be opened shareable for online access the locking rate (record locking) may increase significantly which in turn may influence overall system performance. If you allow online access make sure the locking increase does not decrease your overall system performance.

Since the sample interval for collection profiles that refer to the SNMP extension is limited to 60 sec and less data collected is collected compared to the VSI PERFDAT OpenVMS data collector, online access will not affect system performance in most cases.

- Auto-trend start time
  This field contains the date of the last successful auto-trend engine run. This field is read by the auto-trend engine whenever it is triggered in order to get the start date for data processing. After successful completion the auto-trend engine updates the field. This field can be manually redefined with PERFDAT_MGR. Thus, the user can advise the auto-trend engine to reprocess data.
- Baseline start time
  This field is only accessed by the auto-trend engine, and cannot be changed by the user. The auto-trend engine automatically detects baseline data (for more information about baseline data please refer to VSI PERFDAT Auto Trend Engine) and stores the start time of the baseline data in this field. This is mainly done in order to identify if the baseline data has changed the next time the auto-trend engine is triggered to run. Whenever a new baseline is defined a new data file is created for baseline deviation reports.

- Online performance alerting
  This field defines if the online performance alerting subsystem shall be invoked automatically for the performance data collection started.
- Alert definition file
  This field is read by the online performance alerting subsystem when it is invoked during start-up. It defines the alert definition file to apply to the auto-started performance data collection.
- Data flush time
  Each performance data collection started creates a new data file daily. The time entered at the data flush time prompt defines the time of day a new data file shall be created for the auto-started performance data collection. Enter a valid time string only.

VSI PERFDAT SNMP extension specific fields:

- Agent node
  It defines the OpenVMS node to start the remote collection via SNMP.
- Remote IP address
  It defines the IP address of the remote system that shall be monitored.

Note

Although the user can either enter the fully qualified IP node name or the IP address, it is recommended to enter the IP address in order avoid data loss due to name resolution problems (name server not accessible …)

- SNMP community string
  This field defines the community string to use for the SNMP request. Make sure that the community string defined exists on the remote system. Otherwise update the value according to the remote system. The defaults are
  - Tru64 public
  - Brocade switches FibreChannel
  - Solaris public
  - Linux public

VSI PERFDAT EVA extension specific fields:

- Agent node
  It defines the OpenVMS node to start the HP StorageWorks Virtual Array (EVA) performance data collection
- EVA access device
  It defines the console access device to the EVA (HP StorageWorks Virtual Array) system you want to monitor. You can access the console of an EVA system only if the 'Console LUN ID' parameter of the EVA system is greater than zero. If the 'Console LUN ID' parameter is greater than zero, and you have executed the MCR SYSMAN IO AUTOCONFIGURE command you will get a 1GGAxxx device, where xxx = 'Console LUN ID' parameter value of the EVA system. This is the device name has to be entered.

Collection profile table

The collection profile table contains all predefined collection profiles. Collection profiles are required for starting a data collection. A collection profile defines the sample interval and the metrics to collect. The metrics differ depending on the system to monitor (OpenVMS, HP StorageWorks Virtual Array (EVA) system or any remote system supported by the VSI PERFDAT SNMP extension).

The table is accessed by the VSI PERFDAT OpenVMS data collector, the VSI PERFDAT EVA extension and the VSI PERFDAT SNMP extension during the initialization phase of VSI PERFDAT. During run-time none of the collector processes (PERFDAT,PERFDAT_EVA, PERFDAT_EVA_x, PERFDAT_SNMP, PERFDAT_SNMP_x) access this table, since starting and stopping performance collections are requested by the management utility PERFDAT_MGR. This is done by transmitting the content of the appropriate collection profiles directly to the collection process via its communication mailboxes. For more information see chapter VSI PERFDAT OpenVMS Data Collector and VSI PERFDAT SNMP extension

For more information about collection profiles please refer to chapter VSI PERFDAT_MGR – Management Interface.

License table

Contains all license keys ever applied. Expired licenses are not deleted automatically – they have to be unloaded manually using PERFDAT_MGR.

Record descriptor table

The record descriptor table contains the field descriptors for all metrics available. This table is accessed by the collection processes (see Fig. 2.7) whenever a new data collection file is created (for more information about data file creation please refer to chapter VSI PERFDAT OpenVMS Data Collector, VSI PERFDAT EVA extension and VSI PERFDAT SNMP extension) or you ran the RDB performance data import utility (see chapter Tools & Utilities). The field descriptors of the metrics enabled are read and inserted into the header of the data file. This guarantees that the data of the file are readable regardless of the version of the DQL environment accessing the data or the version of the OpenVMS system of the data file that is stored, the version of the data collector that created the file and the actual definition of the field descriptors in the record descriptor table of the VSI PERFDAT configuration database.

Report profile table

The report profile table contains all predefined trend, capacity and baseline reports. Report creation is profile controlled. Common to all types of report profiles is that these profiles define the metrics, statistics and elements (e.g. an element is a specific device of the DEVICE metric, or a specific process of the

PROCESS metric) to be extracted from a collection database and processed by the auto-trend engine or the DQL$ utility.

Report data files are organized in the same manner as collection data files (see also chapter VSI PERFDAT distributed performance database). The main difference is that the data files are typically much smaller (1:100 or less) and that report files are not processed by the VSI PERFDAT archiving process and consequently not deleted by the environment. The only way to clean up these report files is to delete these files manually.

A report file contains data of a predefined time period. The time period is not freely definable but limited to

- Day
- Week
- Month
- Quarter
- Year
- Unlimited (for baseline reports only)

Whenever the selected time period has expired a new report file is created. E.g. if the time period is WEEK, one report data file per week will be produced.

Three types of reports are configurable:

- Trend report
- Capacity report
- Day to day deviation Report
- Baseline deviation report

Trend report

A trend report compresses the selected statistics of the defined metrics and elements. This means, the selected (defined) data are averaged according to the time compression. The time compression is freely definable, but has to be greater than the sample interval of the collection that created the collection database. Thus, if the time compression is set to 30 min you get 48 values per day and defined statistics in the report profile .The source data has to be collected with a sample interval smaller than 30 min. Trend reports are helpful if the user is interested in detecting changes of the system characteristics over time. E.g. if a trend report is created on a weekly bases, trend report data of week 2 and week 25 can easily be compared with the VSI PERFDAT GUI. From the graphs created by the GUI the user can directly identify if for example the level of CPU load on Monday of week 2 is still the same as on Monday of week 25, or if it has changed and you can easily examine the way it has changed. In addition the user can directly identify if the change in the course of the workload is just limited to specific day of the week, due to any system problem on a specific day, or if a workload change can be identified on each day of week. If the workload has changed on each day of week persistently the system characteristics have changed.

Capacity report

Capacity reports are the basis for capacity planning and forecast analysis. A capacity report contains a daily value per defined statistics. Thus, it is very easy to identify if the workload on a system increases, decreases or remains stable over a long period of time. From a performance point of view not all the daily data are of the same interest. It is common to most systems that there are times that a system is idle and times that a system is busy. For capacity planning purposes it is the busy times that are of interest. Thus, up to 5 different time ranges can be defined for a capacity report to cover these busy times. Only the data within these time ranges are used for calculating the average values.

Note

As stated previously the time period a single report file covers is DAY, WEEK, MONTH, QUARTER or YEAR. It is recommended to define QUARTER or YEAR for capacity reports. If the selected time period is smaller, it is very likely that there is less data within one data file to derive reliable capacity forecasts.

Day to day deviation report

When defining a capacity report profile the user will be asked to enable day to day reporting. Thus, a day to day report is not defined by an extra report profile, but is derived from the capacity report profile. If day to day deviation reporting is enabled for a capacity report the same statistics are used for this report. The day to day report is not added to the capacity report data file, but an extra report is created. The day to day deviation report is derived in a similar manner to the capacity report (averaging the statistics over the time period defined) but does not store the average values, but compares the actual average values to the average values for the previous day and stores the deviation of these average values [%]. Such a report directly highlights on a daily basis if there are significant load changes within the time periods of interest or not.

Baseline deviation report

The baseline deviation report performs, in principle, the same calculations as the day to day deviation report, but does not compare the actual average values to the previous day's values but calculates and stores the deviation between the actual average values and the average values of the same day of the week from the baseline data. Baseline data are a set of logical storage areas (performance collection data files) that cover a full week. The baseline represents a typical week where the system performance was considered 'normal' based on the customer's knowledge and experience. The baseline has to be defined by the system manager by moving the appropriate logical stores areas of a collection database to the directory PERFDAT$DB_SAVE. It does not matter if the data is moved to this directory on the local node or on the archive node. If the baseline does not cover a full week, only the days of the week are processed that the baseline contains the data from. The main difference to the other reports is that

the time period a baseline deviation report file covers is endless. Endless means the report is extended as long as the baseline data is valid. The system manager can invalidate the baseline data by simply copying a new set of data to the PERFDAT$DB_SAVE directory and deleting the old baseline data. In this case the trend engine, which is part of the auto-trend engine and the DQL$ utility detects the next time the baseline report is triggered, that the baseline data has changed and as a consequence creates a new baseline deviation report file.

Stored procedure table

The stored procedure table contains all user defined statistics and the associated parameters configured by the user via the DQL$ utility. User defined statistics are calculated values that can be accessed by all users that are connected to access servers that share the same VSI PERFDAT configuration database as if these statistics are part of a collection database.

There are several reasons for defining user defined statistics. Here are some examples:

- This feature is important in case you want to normalize data. E.g. the statistics for the system wide CPU load collected by the OpenVMS data collector ranges from 0 … 100% * number of CPUs. Thus, if you are monitoring a system with 8 CPUs the statistics for the system wide CPU load collected by the OpenVMS data collector ranges from 0 … 800 %. In order to fetch normalized data of the system wide CPU load ranging from 0 … 100 % a user defined statistics can be created (in this example the user defined statistics is named $iCpuNorm, but you can choose any other name)

  $iCpuNorm = iCpuLoad / iCpuCnt

  where

iCpuLoad	system wide CPU load collected by the OpenVMS data collector
iCpuCnt	number of CPUs.

- You can use this feature to create special statistics you are interested in if these statistics are not directly collected by the OpenVMS data collector, the EVA extension or the SNMP extension but all input parameters to compute them are available. E.g. you are interested in the average I/O size of disk I/Os. The average I/O size is not collected by the OpenVMS data collector but the number of I/Os to the device (iIOs) and the throughput (iMBs) is collected. If you request the data of the user defined statistics

  $iIOSize = iMbs / iIOs

  for a disk device the average I/O size values are returned.

For more information about user defined statistics please see chapterStored procedure engine or the manual VSI PERFDAT – DQL$ Reference manual.

Regional setting table

The regional setting table contains the regional settings defined by the user via the DQL$ utility.

Regional settings define the list separator, the format of numbers, date and time of the CSV files that are mapped, loaded or imported to the distributed VSI PERFDAT performance database as well as how the DQL$ utility formats numbers, date, time and the list separator when exporting performance data to CSV files.

Defining regional settings increases the flexibility to map, load or import CSV files from different sources without any format preprocessing. In addition data can be exported to CSV files using the format expected by the target system to transfer the CSV file.

For more information about the defining and the use of regional settings please refer to the DEFINE REGION, MAP, LOAD, IMPORT and EXPORT command description in the manual VSI PERFDAT– DQL$ Reference manual.

VSI PERFDAT cluster view database

The VSI PERFDAT cluster view database contains all cluster views defined by the user via the DQL$ utility or the GUI.

A cluster view maps performance data of different nodes for cluster wide performance analysis. Once a cluster view is created a virtual collection database is accessible that refers and maps the data of the cluster view members. The advantage is that such a virtual cluster view collection database can be accessed in the same way by the DQL$ utility and the VSI PERFDAT GUI as if it is a collection database created by the OpenVMS data collector, the EVA extension or the SNMP extension. Thus, all methods and features to analyse performance data of single nodes are available for cluster views too. Consequently the workflow to analyse cluster view performance data does not differ from the workflow to analyse single node performance data.

Although in most cases cluster views will be created for cluster wide performance data analysis of OpenVMS clusters there exists no restriction that performance collection databases of OpenVMS cluster members only can be members of a cluster view. Any collection database of any node available can be added to a cluster view. The only restriction is that all collection databases of a cluster view were created with the same sample interval.

One VSI PERFDAT cluster view database exists per node since cluster view definitions are node specific. The file name of the node specific cluster view database has the format:

PERFDAT$CFG:DQL_VIEW_*nodename*.CFG

For more information about cluster view please refer to chapterCluster view engine or to the manual VSI PERFDAT– DQL$ Reference manual.

VSI PERFDAT Query Interface (DQL)

The VSI PERFDAT query interface DQL (Data Query Language) is the common data access layer to the distributed performance database. It provides a data abstraction and a network abstraction layer.

As described in chapter VSI PERFDAT distributed performance database, any data file within the distributed performance database stores all meta-data necessary to access the data in the file header (field and record descriptors, data link descriptors, index reference table descriptor etc.). Due to the fact that the query interface needs no implicit knowledge about the internal structure of the data files, there exists no version dependency when accessing the data. This data abstraction layer guarantees version independency for read access.

The network abstraction layer grants transparent access to any data within the defined community. A community is a logical partition of the whole environment and defines the database view when accessing the data via one system within a community regardless of where the data files are actually stored within the community. Systems of particular interest to a PERFDAT user can be configured in the context of a community. The systems that belong to a particular community is freely definable e.g. all members of a cluster might be part of a particular community, or standalone systems running the same application may be part of another community. The community definition is not cluster bound.

The command syntax of DQL is similar to SQL and makes data query easy.

Query Interface – Prerequisites

TCPIP for OpenVMS has to be installed and configured or any other equivalent and supported TCP/IP product such as TCPware or MultiNet.

Query Interface – Features

- Query interface (DQL) similar to SQL
- Transparent single point access via network abstraction layer
- Up- and downward data compatibility via data abstraction layer
- Dynamic CSV file mapping capability for accessing and analyzing data from different data sources
- Multi file version support
- CSV load capability
- CSV file import capability (data is not only inserted but also normalized)
- CSV export capability
- Statistic package fully integrated in data query interface
- Stored procedures (= user defined statistics).
- Data Clustering (= ability to define cluster views. This feature enables the user to run cluster wide data analysis without any change in the

workflow. All methods and features to analyse performance data of single nodes are available for cluster views too).

Query Interface – Components

The query interface is not a monolithic layer but consists of six components as shown in Fig. 2.8.

- DQL$SRV (DQL server)
- Cluster view engine
- Stored procedure engine
- Statistics Package
- DQL$ command line utility
- PDBC$SRV (Performance database connectivity server)

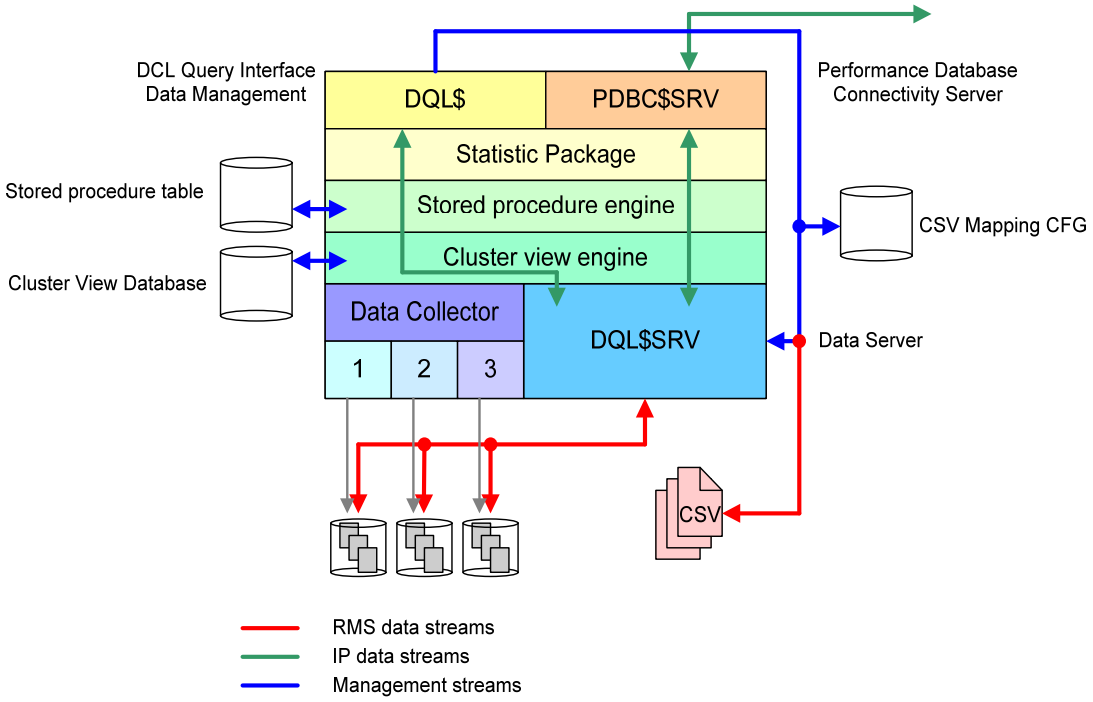


Fig. 2.8 Components of the DQL interface.

DQL$SRV (DQL server)

The DQL$SRV (DQL server) represents the data abstraction layer of the DQL interface. This component directly accesses the data of the performance data stored locally according to the definitions in the header of the data files. Its main task is to map the data query command received from the cluster view engine to RMS calls. Data read from the data files are converted into type independent format and returned compressed to the calling layer. It handles data of the collection databases stored locally as well as CSV files mapped locally. In order to access a CSV file DQL$SRV reads the CSV descriptors from the CSV mapping database (see Fig. 2.8) that defines the layout of the CSV file. The

CSV mapping (inserting the CSV descriptors into the CSV mapping database) has to be done manually by the user using the MAP command of the DQL$ command line utility (see the DQL$ command line utility reference section).

The DQL server is implemented as an IP service. Thus, any node within your environment can request data from the DQL$SRV. Up to 99 DQL$SRV connections are allowed per node. Each DQL$SRV process can access up to 2048 data files concurrently.

The default listener port number of theDQL$SRV service is 3879. It can be redefined by setting the system-wide logical DQL$SRV_PORT. Whenever this logical has been modified the DQL$SRV service has to be restarted using the LAUNCH DQL$SRV command of the PERFDAT_MGR utility.

Note

Once the DQL$SRV_PORT logical has been modified and the DQL$SRV service has been restarted, you have to modify the logicals

- PERFDAT$COMMUNITY
- PERFDAT$ARCHIVE_NODE

on all nodes where these logicals refer the node you have redefined the DQL$SRV listener port. To signal all VSI PERFDAT components that the DQL$SRV listener port on a remote node differs from the default (3879), you have to add the new DQL$SRV listener port to the node name string separated be a semicolon when you define these system-wide logicals.

All informational, warning and error messages during the runtime of the DQL$SRV processes are posted to OPCOM and stored in log-files. The log-files are located in the PERFDAT$LOG directory. The filename has the following format

DQL$SRV_*nodename*.LOG

The DQL$SRV component can be explicitly started by invoking either of the commands listed below, since there exists no dependency to any other VSI PERFDAT SW-component:

- $ MCR PERFDAT_MGR LAUNCH DQL$SRV
- $ @SYS$STARTUP:DQL$SRV_STARTUP.COM

Starting DQL$SRV in stand-alone mode may be important if the local node hosts the distributed performance database or parts of it, but the local nodewill not be used as an access server for the GUI.

If you have not installed TCPIP for OpenVMS on your system but you are using another product such as MultiNet or TCPware please modify DQL$SRV_STARTUP.COM accordingly to add the DQL$SRV IP service.

Cluster view engine

The cluster view engine provides the feature to map performance data of different nodes for cluster wide performance analysis. Once a cluster view is created a virtual collection database is accessible that maps the data of the cluster view members. The advantage is that such a virtual cluster view collection database can be accessed in the same way by the DQL$ utility and the VSI PERFDAT GUI as if it is a collection database created by the OpenVMS data collector, the EVA extension or the SNMP extension. Thus, all methods and features to analyse performance data of single nodes are available for cluster views too. Consequently the workflow to analyse cluster view performance data does not differ from the workflow to analyse single node performance data.

Although in most cases cluster views will be created for cluster wide performance data analysis of OpenVMS clusters there exists no restriction that performance collection databases of OpenVMS cluster members only can be members of a cluster view. Any collection database of any node available can be added to a cluster view. The only restriction is that all collection databases of a cluster view were created with the same sample interval.

Any data query is passed to the cluster view engine. If the data query requests cluster view data, appropriate data queries are created for all members (= collection databases) of the cluster view. These queries are sent to DQL$SRV. The data streams received from DQL$SRV are merged (stacked) and the merged (stacked) data stream is returned to the calling layer. If the data query received contains no cluster view data requests the query is directly bypassed to DQL$SRV.

Cluster views can be configured using the DQL$ utility or the GUI. Cluster view definitions are node specific. Thus, a cluster view can be accessed by those users only that are connected to the distributed VSI PERFDAT performance data via the same node the cluster view was configured. Cluster view definitions are stored in node specific cluster view databases. The file names of the cluster view databases have the format:

PERFDAT$CFG:DQL_VIEW_*nodename*.CFG

The cluster view databases are accessed by the cluster view engine only. Any cluster view configuration request from the DQL$ utility or the GUI is passed to the cluster view engine. It verifies:

- If all collection databases addressed by the cluster view exist and if there exists at least one matching logical storage area within each collection database. With other words, for at least one day performance data have to exist in all collection databases.
- If all collection databases addressed by the cluster view were created with the same sample interval

If one of these checks fail the configuration request is rejected. Otherwise the view definition is stored in the node specific cluster view database and the newly created cluster view is immediately accessible by the user.

Stored procedure engine

The stored procedure engine enables the user to define side (community, node) specific measures (statistics). Such user defined statistics are calculated values and they are created using the DEFINE PROCEDURE command of the DQL$ utility by assigning a function (procedure) to a freely definable statistics name (For more information about defining stored procedures / user defined statistics please refer to manual VSI PERFDAT– DQL$ Reference Manual. of this manual). Statistics collected by the OpenVMS data collector, the EVA extension or the SNMP extension, existing user defined statistics and constant values can be used within the function (procedure) assigned. The supported operators are +, -, * and /.

Any data query is passed to the stored procedure engine. If the data query requests user defined statistics, the data query is modified to request all base statistics necessary to calculate the user-defined statistics. The modified query is passed to the cluster view engine. Once the stored procedure engine receives data from the cluster view engine the user defined statistics is calculated according to the assigned function (procedure) and the result is returned to the caller. In case of stacked requests (SELECT and CALCULATE queries) the input statistics are stacked before calculating the user defined statistics.

User defined statistics are marked with a dollar ($) sign in front to indicate that they are calculated statistics. The user can, but doesn't, have to enter the dollar ($) sign when defining the stored procedure. If the dollar sign is omitted it is automatically assigned.

User defined statistics and the assigned functions (procedures) are stored in the stored procedure table of the VSI PERFDAT configuration database. Thus, once a user-defined statistics has been successfully defined it is immediately accessible by all users accessing data via one of the nodes that share the same VSI PERFDAT configuration database.

The stored procedure table of the VSI PERFDAT configuration database is accessed by the stored procedure engine only. Any stored procedure (user defined statistics) configuration request is passed to the stored procedure engine. It performs several checks before it inserts the user-defined statistics into the stored procedure table of the VSI PERFDAT configuration database:

- It checks if all statistics defined within the function (procedure) assigned to the user defined statistics exist.
- It checks the syntax of the function (procedure) assigned
    - It checks if all brackets are present
    - It checks if supported operators (+, -, *, /) are applied only

If one of these checks fail the configuration request is rejected.

There are several reasons to use this feature. Here are some examples:

- This feature is important in case you want to normalize data. E.g. the statistics for the system wide CPU load collected by the OpenVMS data collector ranges from 0 … 100% * number of CPUs. Thus, if you are monitoring a system with 8 CPUs the statistics for the system wide CPU load collected by the OpenVMS data collector ranges from 0 … 800 %. In order to fetch normalized data of the system wide CPU load ranging from 0 … 100 % a user defined statistics can be created (in this example

the user defined statistics is named $iCpuNorm, but you can choose any other name)

$iCpuNorm = iCpuLoad / iCpuCnt

where

iCpuLoad	system wide CPU load collected by the OpenVMS data collector
iCpuCnt	number of CPUs.

- You can use this feature to create special statistics you are interested in if these statistics are not directly collected by the OpenVMS data collector, the EVA extension or the SNMP extension but all input parameters to compute them are available. E.g. you are interested in the average I/O size of disk I/Os. The average I/O size is not collected by the OpenVMS data collector but the number of I/Os to the device (iIOs) and the throughput (iMBs) is collected. If you request the data of the user defined statistics

  $iIOSize = iMbs / iIOs

  for a disk device the average I/O size values are returned.

Statistics package

Any query is passed to the statistics layer. The query is analyzed if it contains a statistics request. If this is the case appropriate data queries are sent to stored procedure engine. The data received from the stored procedure engine are decompressed, locally cached, processed according to the statistics request and the final result is returned to the caller. If the query is a data query the query is bypassed directly to the stored procedure engine. (For more information about the statistics package please refer to the manual VSI PERFDAT Statistic Package).

PDBC$SRV (Performance data connectivity server)

The performance data connectivity server services data and statistics query sent from the GUI. PDBC$SRV and the DQL$ command line utility represent the network abstraction layer of the DQL interface. Its main tasks are:

- Creating a virtual root file (memory resident) whenever a user connects to the distributed performance database using VSI PERFDAT GUI. This is done by checking the community (PERFDAT$COMMNUITY) definition, establishing connections to DQL$SRV on the nodes listed in the logical PERFDAT$COMMNUITY, the archive node if defined, plus the local node. Once the connection is established PDBC$SRV asks the DQL server to return all known data files. The PDBC$SRV filters these files that belong to the community (data files that are created by the members of the community) and caches the data file links. Thus, the PDBC$SRV keeps the knowledge were the data files are located and how to access.
- Passing the data and statistics queries to the appropriate nodes that host the data files. If the query refers to data files that are stored on different nodes, the performance data connectivity server de-assembles the query, forwards appropriate queries to the nodes, consolidates the data received and returns the result to the caller.

The performance data connectivity server is implemented in a similar manner to the DQL$SRV as an IP service. Up to 99 concurrent PDBC$SRV (PC-client) connections are allowed per node.

The default listener port number of thePDBC$SRV service is 5254. It can be redefined by setting the system-wide logical PDBC$SRV_PORT. Whenever this logical has been modified the PDBC$SRV service has to be restarted using the LAUNCH PDBC$SRV command of the PERFDAT_MGR utility.

All informational, warning and error messages during the runtime of PDBC$SRV processes are posted to OPCOM and stored in log-files. The log-files are located in the directory PERFDAT$LOG. The filename has the following format

PDBC$SRV_*nodename*.LOG

The PDBC$SRV component can be explicitly started by invoking either of the commands listed below, since there exists no dependency to any other VSI PERFDAT SW-component:

- *$ MCR PERFDAT_MGR LAUNCH DQL$SRV*
- *$ @SYS$STARTUPDQL$SRV_STARTUP.COM*

Starting PDBC$SRV in stand-alone mode may be important if the local node is be used as an access server (GUI) only, and no collection data files are stored on the node.

If you have not installed TCPIP for OpenVMS on your system but you are using another product such as MultiNet or TCPware please modify PDBC$SRV_STARTUP.COM accordingly to add the PDBC$SRV IP service.

DQL$ command line utility

The DQL$ command line utility is like the performance data connectivity server responsible for transparent access to the data files within the defined community (network abstraction). As with PDBC$SRV services GUI requests DQL$ services interactive requests from the DCL command line. To invoke the DQL$ command line utility enter

$ MCR DQL$ [/REGION=regional setting]

at the DCL prompt. The /REGION command qualifier can be applied. It defines the default regional setting of the DQL$ session. If this command qualifier is omitted the default regional setting stored in the regional setting table of the VSI PERFDAT configuration database is used (for more information about regional settings please refer to chapter Regional Setting table of this manual or to the manual VSI PERFDAT– DQL$ Reference manual).

Note

Enter the /REGION command qualifier blank separated right after the image activation MCR DQL$. Otherwise the qualifier is not passed to the DQL$ image and the regional setting will not be changed.

The DQL$ command set consists of four main groups

- Data query commands
- Statistics query commands
- Report extraction command
- Data content management commands

You can execute DQL$ command scripts using the @ command. A command script can be any text file that contains valid DQL$ commands.

Data query commands

Table 2.1 summarizes the data query commands available. For more detailed information about the available commands please refer to the online help of the DQL$ command line utility or to the VSI PERFDAT– DQL$ Reference Manual.

Table 2.1 Data query command summary table

Command	Description
ATTACH	Attach a physical or logical storage area or a whole collection database of the distributed performance database.
DEATTACH	Disconnect from a physical or logical storage area or a collection database of the distributed performance database.
DEFINE HEADER	This command can be applied in advance of the EXPORT and the CREATE GRAPH command in order to enter a user defined header line for the CSV file or a user defined caption for the graph created.
INSERT	Insert fields of a record or the whole record into an existing metric of a physical storage area
SET TRANSACTION	Set the transaction access for a physical or logical storage area or a whole collection database of the distributed performance database. The transaction access can be • READ ONLY (default) • READ WRITE
EXPORT	Exports 1…n statistics from a metric of attached physical or logical storage areas or whole collection databases to a CSV file.
SELECT	Reads data fields from a metric of attached physical or logical storage areas or whole collection databases and displays the data on screen.
CREATE GRAPH	Reads data fields from a metric of attached physical or logical storage areas or whole collection databases, creates graphs from the data, converts these graphs into PNG format and stores them either in a user defined directory or in the directory PERFDAT$GRAPH.

Statistics query commands

Table 2.2 summarizes the statistics query commands available. For more detailed information about the available commands please refer to the online help of the DQL$ command line utility or to the VSI PERFDAT– DQL$ Reference Manual.

Table 2.2 Statistics query command summary table

Command	Description
DEFINE ELEMENT	This command can be applied in advance of the stacked form of CALCULATE command in order to assign a user defined element name to the stacked element of a stacked calculation or deviation report.
CALCULATE	This command is used for two different types of calculations • Depending on the parameters applied, this command calculates the arithmetic mean value, integral mean value, max value, standard deviation or all of this for 1...n statistics of a metric. • Deviation analysis 1...n statistics of a metric are read from two different logical storage areas (= data of different days). These data are averaged and compared to each other. The percentage the source data average differs from the reference data average is displayed for each statistics and element. The deviation analysis can be done integral or arithmetic based.
CORRELATE	Calculates the correlation between different statistics of different metrics within a logical storage area.

Report extraction command

Table 2.3 describes the report extraction command in brief. For more detailed information about the available commands please refer to the online help of the DQL$ command line utility or to the VSI PERFDAT– DQL$ Reference Manual.

Table 2.3 Report extraction command summary table

Command	Description
EXTRACT	The EXTRACT command is used to create reports (trend, capacity and baseline reports) according to predefined report profiles. With the EXTRACT command you can apply any predefined report profile to any collection database. The only restriction is that the report is of the same type as the collection database (OpenVMS reports can only be applied to OpenVMS collection databases, Tru64 reports can only be applied to Tru64 collection databases and so on).

Data content management commands

Table 2.4 summarizes the data content management commands available. For more detailed information about the available commands please refer to the online help of the DQL$ command line utility or to the VSI PERFDAT– DQL$ Reference Manual.

Table 2.4 Data content management command summary table

Command	Description
CHECK	This command checks if the CSV files addressed by the CSV mapping entries in the CSV mapping database are valid.
CREATE	Depending on the parameters applied the CREATE command is used for creating a new physical storage area or a new metric (table) within an existing physical storage area.
CONVERT	Converts the header(s) of physical storage areas created by an older VSI PERFDAT version than actually used to new format. Collection databases have to be converted to actual format if data shall be inserted. For read access there is no need to convert the collection databases.
DEFINE	• DATA HOST The DEFINE DATA HOST command defines the node (data host) were new data files created during the current DQL$ session will be stored. The user can define any community member or the archive node as the new data host. • ELEMENT When a stacked element report or a stacked deviation report is created these reports contain the calculated values for a single (stacked) element. This stacked element name can be (re)defined by applying the DEFINE ELEMENT command in advance of the CALCULATE statement.

- HEDAER

  If data are exported to a CSV file the header line (comment) of that file and the caption of a graph created by applying the CREATE GRAPH command can be user defined. This is done by applying this command in advance of the EXPORT and CREATE GRAPH command.

- PROCEDURE

  Defines side specific, calculated statistics (measures) that can, once defined, be accessed as if they are part of the associated metrics of the collection databases available.

- REGION

  Used to define regional settings. The feature to define regional settings increases the flexibility to import, load and mapping CSV files of different format and to export data to a CSV file formatted as expected by the system the CSV will transferred to.

- VIEW

  Creates a cluster view. A cluster view maps performance data of different nodes for cluster wide performance analysis.

DROP

Depending on the parameters applied the DROP command deletes a metric within a physical storage area, a physical storage area, a logical storage area or a whole collection database.

IMPORT

Imports data of a CSV file into an existing collection database. Prerequisite for importing data of a CSV file is a valid descriptor file for the CSV file, and the CSV file has to contain a time column. For more information about creating a CSV descriptor file please refer to VSI PERFDAT– DQL$ Reference manual

Using the IMPORT command CSV data are normalized before they are inserted into the collection database. It is very likely that the timestamps in the time column do not match the timestamps in the collection database. This is a prerequisite when correlating data. Any correlation based on data that does not match in time (timestamp of a sample, sample interval) will return wrong results. Normalizing means that based on the CSV data expectancy values are calculated for the timestamps of the collection database. An integral based algorithm is used to normalize the data.

LIST

- METRIX

  This command displays all the performance metrics (tables) stored in the record descriptor table of the VSI PERFDAT configuration database.

- STATISTICS

  The LIST STATISTICS command displays the statistics stored in the record description table of the VSI PERFDAT configuration database and the user defined statistics (stored procedures) of a particular performance metric (table). The field name, data type, field length and the field description is displayed.

LOAD

Loads data of a CSV file into an existing collection database. Prerequisite for importing data of a CSV file is a valid

descriptor file for the CSV file, and the CSV file has to contain a time column. For more information about creating a CSV descriptor file please refer to VSI PERFDAT– DQL$ Reference manual.

Using the LOAD command CSV data are not pre-processed (normalized) before they are inserted into the collection database. The LOAD command should only be used if the timestamps in the CSV file match exactly the timestamps in the collection database. Otherwise it is recommended to use the IMPORT command. The LOAD command consumes less system resources and is faster than the IMPORT command.

MAP

Maps CSV files to the distributed performance database. Mapping a CSV files means that the data of the CSV file can be accessed as if they are part of a collection database. Prerequisite for mapping CSV files are

- In the first row of the CSV file the nodes the data shall be visible to are inserted as a comma separated list.
- The second row has to contain the column headers. Max length of a header item is 64 characters. Max number of columns is 200.
- One column header item has to be named 'Time' and the data format of that column has to be OpenVMS date/time format.
- The data rows of the CSV file have to be ordered descending by the 'Time' column.
- A CSV descriptor file that contains a valid CSV record descriptor. For more information about creating a CSV descriptors and CSV descriptor files please refer to VSI PERFDAT– DQL$ Reference manual.

It does not matter if the CSV file includes data of different days. DQL splits the file virtually in order to map the CSV file content to the database scheme.

There may co-exist several rows with the same time-stamp. In that case 1 to max 3 columns can be defined as index fields in the CSV descriptor file. It is not allowed to have duplicates in the CSV file. A duplicate record contains the same the timestamp and the index fields contain the same data as another record.

Mapped CSV files can be accessed read only.

CSV data content cannot be correlated to other CSV file content or to data of a collection database.

CSV file mappings are only valid on the node where the mapping is done but the CSV content can be accessed by any member of the community the node that hosts the CSV file belongs to in case the node(s) listed in the first line of the CSV file are also member(s) of the community.

Mapped CSV files are not managed by the VSI PERFDAT environment. They have to be managed by the system manager.

In order to map CSV file content a record descriptor is required to define the record layout of the CSV file(s). CSV record descriptors are stored in so called CSV descriptor files. The CSV record descriptor file is inserted into the CSV mapping database. PERFDAT$CFG:CSV_PROFILES.CFG when the MAP command is executed. Thus, regardless if the CSV

descriptor file is deleted afterwards, the CSV mapping will be valid till the CSV mapping is manually removed from the CSV mapping database.

REMOVE

- CSV File Mapping

  Removes a valid CSV mapping from the CSV mapping database.

- PROCEDURE

  Removes particular user defined statistics from the stored procedure table of the VSI PERFDAT configuration database.

- REGION

  Removes existing regional settings from the regional setting table of the VSI PERFDAT configuration database.

- VIEW

  Removes an existing cluster view.

SET

- REGION

  Changes the default regional setting for the current DQL$ session and all subsequent DQL$ sessions started on any node that share the same VSI PERFDAT configuration database.

SHOW

Depending on the parameters applied one receives the information listed below

- DATABASE

  Shows all collection databases accessible within the community.

- LOGICAL STORAGE AREA

  Shows all logical storage areas within a particular collection database.

- PHYSICAL STORAGE AREA

  Depending on the parameter applied, the physical storage areas within a logical storage area or within a whole collection database are displayed.

- HEADER

  Shows the headers of all attached physical storage areas.

- METRIC

  Shows the metrics (tables) available from each attached physical storage area.

- STATISTICS

  Shows the statistics available of a particular metric.

- ELEMENT

  Shows all elements of a particular metric stored in all physical storage areas attached. Elements can be sorted by any field of the metric to get hot element statistics easily.

- CSV File Mapping

  Shows the CSV mappings configured on the local node.

- PROCEDURE

	Displays the user defined statistics stored in the stored procedure table of the VSI PERFDAT configuration database. • REGION Displays the default regional setting of the current DQL$ session and all regional settings defined in the regional setting table of the VSI PERFDAT configuration database. • VERSION The SHOW VSERION command displays the version of the DQL$ utility and DQL$SRV of the node the current DQL$ session is connected to • VIEW Displays the cluster views configured on the local node (content of the local cluster view database).
UPDATE	• HEADER The UPDATE HEADER command can be applied to modify any header attributes of a logical storage area or a whole collection database.

Query Interface – Data Flow

The DQL query interface provides transparent access to the distributed performance database. The view of the performance database the user has is defined by the content of the logical PERFDAT$COMMUNITY defined on the access server (for more details see chapter VSI PERFDAT Environment in this manual).

This chapter illustrates the data flow and the effect of the community definition on the database view.

Let us assume the VSI PERFDAT environment consists of 4 nodes – node A, node B, node C and an archive node. The community settings on these nodes differ:

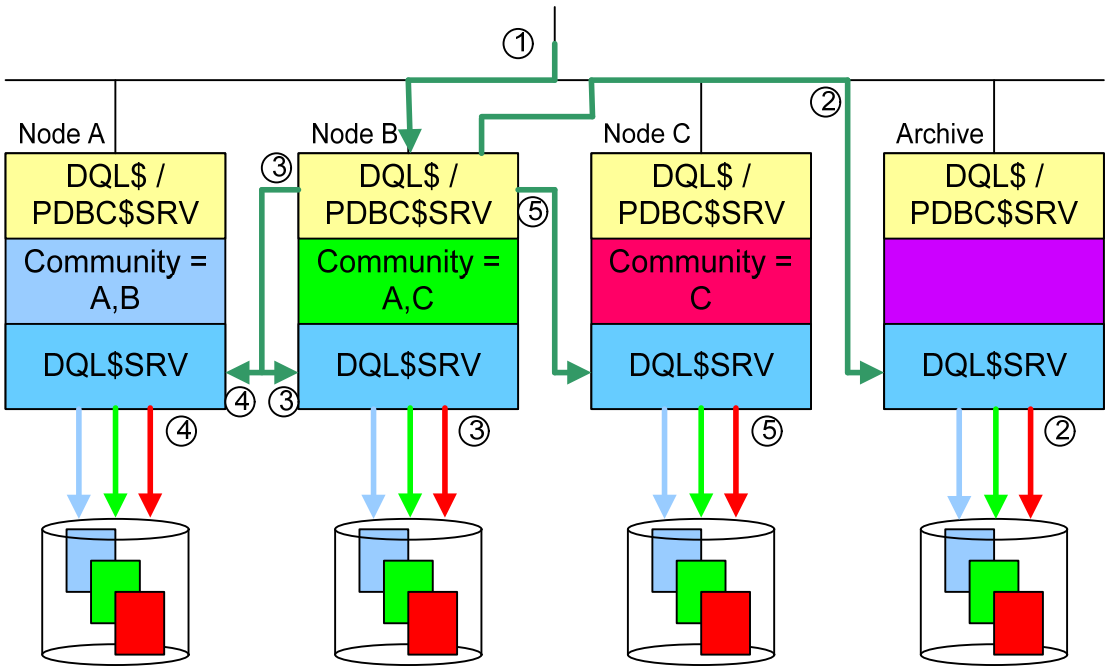
- Node A: PERFDAT$COMMUNITY = Node A, Node B
- Node B: PERFDAT$COMMUNITY = Node A, Node C
- Node C: PERFDAT$COMMUNITY = Node C

Before discussing different access examples let us recap the rules for database views

- All collection databases files that were originally created by the OpenVMS data collector, the EVA extension or SNMP extension running on any node defined by the logical PERFDAT$COMMUNITY and that are stored on any of the community members or on the archive node are visible and accessible via the DQL interface.
- Regardless of the definition of PERFDAT$COMMUNITY all collection databases created by the access server and stored on the access server or on the archive node are visible and accessible via the DQL interface. In other words the access server is always member of the community even if it is not listed in the PERFDAT$COMMUNITY logical.
- Regardless of the definition of PERFDAT$COMMUNITY on the archive node, accessing the archive node provides a database view that consists of all collection databases stored on the access server.

Example 1

Database connect request to Node B



①...⑤ Data Flow, step 1 to 5

Description:

Step1

PDBC$SRV on node B receives the database connect request from the GUI. It checks the community definition and checks if an archive node is defined. Since the access server is always a member of the community regardless of the definition of the logical PERFDAT$COMMUNITY the community resolved by PDBC$SRV consists of

- Node A (defined by PERFDAT$COMMUNITY)
- Node B (not defined by PERFDAT$COMMUNITY)
- Node C (defined by PERFDAT$COMMUNITY)

Step 2

PDBC$SRV connects to DQL$SRV on the archive node first. The DQL server returns all known collection databases using descriptors. Since the community consists of all nodes in the environment all collection database descriptors received are accepted by PDBC$SRV.It adds the collections database links to its database view structure.

Step 3

PDBC$SRV connects to DQL$SRV locally and adds all collection database links returned to its database view structure (remember, the community consists of all nodes – A, B, C). If PDBC$SRV detects a duplicate entry

(the same physical storage areas exists locally and on the archive node) the new (locally received) collection database link is dismissed.

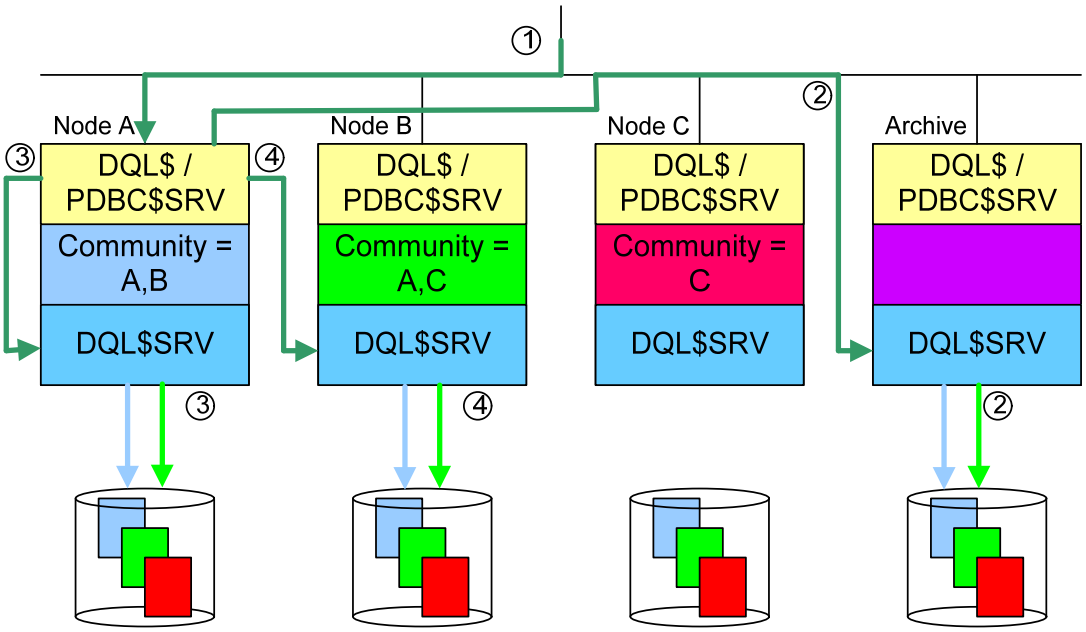
Step 4, 5

Finally PDBC$SRV connects to the node A and C. The collection database descriptors returned are processed in the same way as described in Step 3.

Thus, in this example, the whole distributed performance database of the VSI PERFDAT environment is visible and accessible to the user.

Example 2

Database connect request to Node A



①...④ Data Flow, step 1 to 4

Description

Step1

PDBC$SRV on node A receives the database connect request from the GUI. It checks the community definition and checks if an archive node is defined. In this example the node list of the logical PERFDAT$COMMUNITY contains node A and node B. Since the access server is already included in the node list defined by PERFDAT$COMMUNTY the community resolved by PDBC$SRV consists of

- Node A (defined by PERFDAT$COMMUNITY)
- Node B (defined by PERFDAT$COMMUNITY)

Step 2

PDBC$SRV connects to DQL$SRV on the archive node first. The DQL server returns all known collection databases using descriptors. Since the community consists of node A, B only these collection database descriptors received are accepted by PDBC$SRV and added to its database view structure that refer to node A and B.

Step 3

PDBC$SRV connects to DQL$SRV locally, and adds all collection database links returned to its database view structure that refer to node A an B. If PDBC$SRV detects a duplicate entry (the same physical storage areas exists locally and on the archive node) the new (locally received) collection database link is dismissed.

Step 4

Finally PDBC$SRV connects to the nodes. The collection databases descriptors returned are processed the same way as described in Step 3.

Thus, in this example, a selective view of the distributed performance database consisting of node A and B is visible and accessible to the user.

Online Performance Alerting Subsystem

The online performance alerting subsystem provides real-time alerting capabilities. Online performance alerting can be enabled for any active performance data collection, independently if the data collection is performed by the OpenVMS data collector, the SNMP extension, the EVA extension or the VSI PERFDAT API.

Once online alerting has been enabled for an active performance data collection the alerting subsystem tracks the actual values of specific statistics collected by the OpenVMS data collector,the SNMP extension, the EVA extension or by use of the VSI PERFDAT API and triggers alerts if any alert condition becomes true or it triggers an alert clear message (informational) if an alert condition is no longer true.

The statistics to monitor, the alert conditions and the alert method are defined by alert blocks within an alert definition file. Thus, prerequisite for enabling online alerting for an active performance data collection is that a valid alert definition file exists. An alert definition file is a text file for easy customization.

Any statistics collected by the active collection online alerting has been enabled as well as valid use defined statistics can be configured to be monitored. For more information about user defined statistics please refer tochapter Stored procedure engineor to the manual VSI PERFDAT– DQL$ Reference manual.

The maximum number of elements tracked by the online performance alerting subsystem per alert block is 4096. If the number of elements associated with an alert block exceeds the supported number of elements the online performance alerting subsystem through a warning OPCOM message.

The online performance alerting subsystem is invoked by the OpenVMS data collector, the VSI PERFDAT EVA extension, the VSI PERFDAT SNMP extension and the VSI PERFDAT API either automatically during start-up if auto-start entries are configured for online performance alerting or on user request via the PERFDAT_MGR utility.

For detailed description of alert definition file and alert blocks please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference Manual.

Once the online performance alerting subsystem is triggered for an active performance data collection, it checks if the alert definition file contains valid alert blocks. If this check fails the calling process gets informed and online performance alerting will be disabled for the data collection. (Thus, any error detected by the online performance alerting subsystem is logged in the callers log file.)

If the alert definition context check succeeds the online performance alerting subsystem starts monitoring the statistics defined by the alert blocks.

The available alert methods to inform system management about alert conditions are:

- OPCOM messages (always)
- User definable command scripts to be executed on warning or critical conditions. These command scripts are defined within the alert blocks of the alert definition file. (see VSI PERFDAT– PERFDAT_MGR Reference Manual – ENABLE ALERT command)

Warning alert OPCOM messages always start with PERFDAT-W-ALERT – critical alert OPCOM messages always start with PERFDAT-E-ALERT.

Each OPCOM message contains full information about the alert condition:

- Severity (PERFDAT-E-ALERT, PERFDAT-W-ALERT, PERFDAT-I-ALERT)
- Node name the alert was triggered
- Element that exceeds the warning/critical threshold
- Statistics that exceeds the warning/critical threshold
- Metric the statistics belongs to
- Value of the statistics
- Threshold value
- Comparison operator (LT. EQ, GT)

These parameters are also passed to the user definable command scripts.

Each alert block is processed independently by the online alerting subsystem. When the alerting subsystem processes an alert block it opens a warning alert content and a critical content file for the alert block in the directory PERFDAT$ALERT. The file name format of these content files is

- ERRMSG_*metric*_*nn*_*node*.COM

where *metric* is the metric defined by the METRIX parameter keyword of the alert block, *nn* is a number and *node* is the local node name. Depending if a warning or a critical alert condition is triggered a new record is stored in either of this file containing the appropriate user defined alert command script and the parameters (P1 – P7) as defined below:

- P1 Node name
- P2 Metric Name
- P3 Statistics name
- P4 Element name
- P5 Average value of the statistics
- P6 Threshold values
- P7 Severity level (1=clear, 2=warning, 3= error)

When the alert processing has completed for the alert block these alert content files are closed. If these alert content files contain valid records they are queued to the batch queue defined herein. If these files contain no records the files are unconditionally deleted, otherwise they are deleted after the batch job has completed successfully. If the batch job fails these files are kept. Thus one can check if all alerts have been submitted by checking the directory PERFDAT$ALERT for files named ERRMSG*.COM.

Independently of the alert method defined in the alert definition file all alerts are written to log files. The log filesare stored in the directory PERFDAT$ALERT. The log file that contains the OpenVMS alerts has the format:

PERFDAT_ALERT_*node*.LOG_*date*

The format of the log file that contains SNMP extension alerts is:

PERFDAT_SNMP_x_ALERT_*node*.LOG_*date*

where:

- *node* local node name
- *date* date the log file was created
- x working process number of the SNMP extension

The format of the log file that contains EVA extension alerts is:

PERFDAT_EVA_x_ALERT_*node*.LOG_*date*

where:

- *node* local node name
- *date* date the log file was created
- x working process number of the EVA extension

The format of the log file that contains VSI PERFDAT API alerts is:

*application*_ALERT_*node*.LOG_*date*

where:

- *application* Application name
- *node* local node name

On day change a new log file is created if online alerting has been enabled for any data collection managed by the caller of the online performance alerting sub-system (OpenVMS data collector, SNMP extension, EVA extension, VSI PERFDAT API). Thus, one log file contains all alerts triggered by the alert subsystem on that day.

Performance database file name cache service DQL_NAME

General description

The distributed performance database is organized in way that there exists no persistent root file for any VSI PERFDAT performance database on disk (see chapter VSI PERFDAT distributed performance database). All meta-data (field and record descriptors, data link descriptors, index reference table descriptor …) necessary to access the data is stored in the header of each physical storage area. The advantage is that data files can be moved to any OpenVMS node and the data file stays read accessible without any additional actions such as data conversion, unload and load operations. On the other hand the meta-data have to (re)fetched any time a user connects to the distributed performance database via the DQL$ utility or the GUI in order to create a virtual root file (data link cache) required to access the data.

Prior to VSI PERFDAT V3.2 the meta-data were fetched by performing a full data file header scan on all members of the PERFDAT community. Thus, prior to VSI PERFDAT V3.2 the initial connect request to the distributed performance database took few seconds up to minutes.

In order to solve that performance issue VSI PERFDAT provides the performance database file name cache service DQL_NAME first introduced with VSI PERFDAT V3.2.

The performance database file name cache service DQL_NAME provides a database file name cache to all VSI PERFDAT components that contains full header information about all VSI PERFDAT database files locally stored. As long as the performance database file name cache service DQL_NAME is available and the database file name cache is marked valid all VSI PERFDAT components obtain database file header information from that cache rather than scanning the files on disk. Thus, the initial connect request speeds up dramatically (ten times and more) compared to VSI PERFDAT V3.1 and priorreleases of VSI PERFDAT.

The file name cache is updated by the DQL_NAME process:

- periodically once per TTL (time to leave) duration.
- whenever a VSI PERFDAT component creates, renames or deletes a database file.

The TTL (time to leave) parameter defines the time duration the entries in the performance database file name cache are valid. Every entry has to be updated once during the TTL duration by DQL_NAME process. If TTL duration time expires and the entries in the cache have not been updated for any reason the cache is marked invalid and from this time on all VSI PERFDAT components will fetch the file header information direct from disk on connect requests until the performance database file name cache service DQL_NAME starts (re)processing again. Thus, the TTL parameter defines the cache entry life-time.

The default value of the TTL parameter is 30 minutes.

Note

If you delete any VSI PERFDAT data file in the directory PERFDAT$DB manually the file name cache will not be updated automatically until TTL time expires. Thus, in this case you have to trigger a rebuild of the performance database file name cache manually using the PERFDAT_MGR utility to keep the cache up to date. If you run VSI PERFDAT in a cluster you have to rebuild the file name cache on all cluster members.

The performance database file name cache service DQL_NAME is managed and controlled via the PERFDAT_MGR utility.

Startup / Shutdown

The performance database file name cache service DQL_NAME can be started either via the PERFDAT_MGR utility (see chapterPERFDAT_MGR Management Interface) or directly from the DCL command line using the start-up script

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

The privileges required to start the performance database file name cache service DQL_NAME are:

- CMKRNL
- NETMBX
- OPER
- SYSLCK
- SYSPRV
- TMPMBX
- WORLD

All informational, warning and error messages during runtime are stored in the performance database file name cache service log-file. The log-file is located in the PERFDAT$LOG directory. The filename of the log-file has the format

DQL_NAME_*nodename*.LOG

E.g. the file PERFDAT$LOG:DQL_NAME_BCSXTC.LOG is the log-file of the performance database file name cache service DQL_NAME running on node BCSXTC.

In order to shutdown the performance database file name cache service DQL_NAME enter

```
$ MCR PERFDAT_MGR STOP NAME_SERVER
```

You do not have to start the performance database file name cache service DQL_NAME manually after restarting the VSI PERFDAT environment/DQL

interface. This is automatically done by the VSI PERFDAT and DQL startup routines.

The performance database file name cache service DQL_NAME is automatically shutdown when the whole VSI PERFDAT environment using the command

```
$ MCR PERFDAT_MGR SHUTDOWN ALL
```

is shutdown.

VSI PERFDAT Statistic Package

The statistic package is a part of the query interface as described in the chapter VSI PERFDAT Query Interface (DQL). Thus, it is available from the GUI as well as from the command line interface (DQL$) on OpenVMS.

Any query is passed to the statistics layer. The query is analyzed if it contains a statistics request. If this is the case appropriate data queries are sent to DQL$SRV. The data received from DQL$SRV are decompressed, locally cached, processed according to the statistics request and the final result is returned to the caller. The advantage of having the statistic package implemented server based is that no massive data transfer between the access server and the PC-client running the GUI is necessary to receive the results. Thus, network load does not increase due to statistic queries. In addition, it is guaranteed that the runtime of a statistic query is (almost) the same independent of the location (it does not really matter if the server is located locally or 100 miles away) of the servers and the bandwidth of the network.

The statistical functions implemented are listed below

- Min/max calculations
- Mean value calculations
- Standard deviation
- Correlation
- Integral and mean value based deviation calculation
- Elements can be ordered by any statistics of the metric. This means that the elements are displayed in ascending or descending order based on the percentage of the overall load defined by the statistics caused by each element. The time range is freely definable.

These statistical methods are very helpful for getting a quick overview what is going on the systems without expert knowledge and will – in most cases - reduce analysis time.

VSI PERFDAT Archiving and Housekeeping

VSI PERFDAT provides automatic data management capabilities. These data management activities are performed by the archiving process PERFDAT_ARCHIVE. Once configured, this process carries out the archiving and housekeeping tasks reliable and unattended. All tasks the process performs are listed below in order

- Moving data collection files to PERFDAT$DB_ARCHIVE locally
  Moving any closed data file from the working directory of the OpenVMS data collector, the VSI PERFDAT EVA extension and the VSI PERFDAT SNMP extension – PERFDAT$DB_LOCAL to the PERFDAT$DB_ARCHIVE directory.
- Moving data collection files to the archive node
  All logical storage areas located in PERFDAT$DB_ARCHIVE will be moved to PERFDAT$DB_ARCHIVE on the archive node, if an archive node is defined and if the local node is not itself the archive node. The system-wide logical PERFDAT$ARCHIVE_NODE refers to the archive node. The value of this logical contains the node name of the archive node and the listener port of the DQL$SRV service running on the archive node separated by a semicolon. If it is not defined or if it is set to NOTDEF it indicates to the archiving process that no archive node is configured. If the value of the logical is the name of the local node, the local node is identified as the archive node.
- Data collection file cleanup
  All data collection files in PERFDAT$DB_ARCHIVE older than the actual time minus the keep time are unconditionally deleted in order to save disk space.
- Log-file and temp file cleanup
  All SW-components of the VSI PERFDAT environment create log-files when started. All Log-files in the directories PERFDAT$LOG and SYS$STARTUP created by VSI PERFDAT components are purged with the PURGE Command Qualifier/KEEP=5. That means that the last five versions will always be available for examination, if necessary.
- Triggering the auto-trend engine
  After all data management activities listed above have completed the auto-trend engine is triggered to run.
- Calculating the next time to run the archiving and cleanup jobs

The archiving process performs these tasks once a day. The user can configure the following:

- Enable / disable archive processing
- Time of day that the archive processing starts
- Number of days that old performance data files are kept in the PERFDAT$DB_ARCHIVE directory.

These control parameters are stored in the archive control parameter table of the VSI PERFDAT configuration database as well in a volatile control table. Thus, you can change the parameter during run-time without affecting the default settings stored in the archive control parameter table.

The parameters in the volatile archive control table are changed using the SET command. The parameters of the permanent archive control table are changed using the DEFINE command of PERFDAT_MGR. For more information about the SET and DEFINE command please refer to chapter VSI PERFDAT_MGR – Management Interface or to the online help of PERFDAT_MGR.

Startup / Shutdown

The archiving process is implicitly started (restarted) whenever the OpenVMS performance data collector, the EVA extension or the SNMP extension is started (restarted). You can explicitly start the archiving process with

$ MCR PERFDAT_MGR START ARCHIVE

or

$ @SYS$STARTUP:PERFDAT_ARCHIVE$STARTUP.COM

When the archiving process is (re)started it reads the archive control table of the VSI PERFDAT configuration database, fills the volatile database accordingly and starts processing.

To stop the archiving process use

$ MCR PERFDAT_MGR STOP ARCHIVE

Note

Stopping the archiving process also prevents the auto-trend engine from running, since the archiving process triggers the auto-trend engine. If you just want to stop the archive processing but not the auto-trend engine, then disables the archiving process via PERFDAT_MGR but do not stop the process.

Command procedures

Some tasks are not directly performed by the archiving process but executed by a command script in a sub-process spawned by PERFDAT_ARCHIVE. Thus the tasks executed by the command scripts can be easily modified according to you requirements. The archiving tasks performed by command scripts are listed below:

- Moving data collection files to PERFDAT$DB_ARCHIVE
  - PERFDAT$BIN:PERFDAT_MOVEFILES.COM
- Moving data collection files to the archive node and data collection cleanup
  - PERFDAT$BIN:PERFDAT_ARCHIVE.COM
    - E.g. if you want to use DECnet copy instead of FTP copy replace the COPY/FTP command and update the $STATUS check in the next line.
- Log- and temp-file cleanup
  - PERFDAT$BIN:PERFDAT_HOUSEKEEPING.COM

Archive node

If you do not change PERFDAT$BIN:PERFDAT_ARCHIVE.COM to use other network protocols or TCP/IP products you have to define the anonymous directory on the archive node after installing VSI PERFDAT manually:

$DEFINE/SYSTEM/EXEC TCPIP$FTP_ANONYMOUS_DIRECTORY PERFDAT$DB_ARCHIVE

Make sure that the logical PERFDAT$ARCHIVE_NODE is defined on the archive node and that it refers to its node name.

VSI PERFDAT Auto Trend Engine

The auto-trend engine extracts

- Trend reports
- Capacity reports
- Baseline reports

from the performance collection files without any user action. Prerequisites for running the auto-trend engine are

- The archiving process is started, since it triggers the auto-trend engine.
- All reports are profile controlled. Thus, one or more valid report profiles have to exist in the report profile table of the VSI PERFDAT configuration database.
- A minimum of one of these report profiles has to be enabled for processing by the auto-trend engine.
- Reports are only created for the nodes registered in the auto-start table of the VSI PERFDAT configuration database. As described in chapter VSI PERFDAT configuration database one of the parameters per entry is the collection profile to use for automatic collection start-up. The auto-trend engine extracts the reports from the collection data files created by these collection(s).

The report profiles are created via the PERFDAT_MGR management utility. For more information about creating report profiles and the different types of profiles available please refer to chapter VSI PERFDAT configuration database and PERFDAT_MGR – Management Interface.

Trend, capacity and baseline reports are only extracted from performance data collected on the local node (OpenVMS data collector) or from SNMP and EVA performance data collections that are processed by the local node (node = EVA/SNMP agent node, “Agent name” parameter of the entries in the auto-start table – for more information see chapter VSI PERFDAT configuration database).

VSI PERFDAT_MGR – Management Interface

The VSI PERFDAT_MGR utility is the common management interface for the components of VSI PERFDAT.

The VSI PERFDAT_MGR image PERFDAT_MGR.EXE is located in the directory SYS$COMMON[SYSEXE].

The main tasks of PERFDAT_MGR

- Startup / shutdown of the VSI PERFDAT environment
- Controls and monitors the status of OpenVMS performance data collections
- Controls and monitors the status of performance data collections processed by the PERFDAT EVA extension
- Controls and monitors the status of remote performance data collections via the VSI PERFDAT SNMP extension
- Controls and monitors the status of application data collections processed by the VSI PERFDAT API
- Management /control of the performance data archiving
- Management / maintenance of the VSI PERFDAT configuration database
- Online performance alert management
- Manages / controls the performance database file name cache service DQL_NAME

To invoke PERFDAT_MGR type at the DCL prompt

$ MCR PERFDAT_MGR

Startup / shutdown of the VSI PERFDAT environment

Any privileged user can start and shutdown the whole VSI PERFDAT environment or parts of it via PERFDAT_MGR. Table 2.2 lists the PERFDAT_MGR commands available for start-up and shutdown processing.

Table 2.2 Command reference table for start-up and shutdown processing

Command	Description

LAUNCH ALL	Starts up the whole PERFDAT environment on the local OpenVMS node. The components started are
	• OpenVMS data collector
	• EVA extension
	• SNMP extension
	• All components of the DQL interface
	  ○ DQL$SRV
	  ○ PDBC$SRV
	• Performance database filename cache service
	• Auto Archiving process

LAUNCH PERFDAT	All components but the SNMP extension and the EVA extension will be started as listed for the LAUNCH ALL command.
LAUNCH PERFDAT_EVA	EVA extension startup. All components but the SNMPextension and the OpenVMS data collector will be started as listed for the LAUNCH ALL command.
LAUNCH PERFDAT_SNMP	SNMP extension startup. All components but the EVAextension and the OpenVMS data collector will be started as listed for the LAUNCH ALL command
LAUNCH DQL$SRV	The DQL$SRV will be started only.
LAUNCH PDBC$SRV	The PDBC$SRV will be started only.
START ARCHIVE	Starts up the auto archiving process.
STOP ARCHIVE	Shutdown of the auto archiving process only.
START NAME_SERVER	Starts up the performance database filename cache service.
STOP NAME_SERVER	Shutdown of the performance database filename cache service
SHUTDOWN ALL	Actions performed • Shutdown of the archiving process. • Shutdown of the performance database filename cache service • Stops all active collections of the OpenVMS data collector • Stops all active collections of the SNMP extension • Stops all active collections of the EVA extension • Shuts down the OpenVMS data collector process • Shuts down the SNMP extension master process • Shuts down the EVA extension master process
SHUTDOWN PERFDAT	Actions performed • Stops all active collections of the OpenVMS data collector

	• Shuts down the OpenVMS data collector process
SHUTDOWN PERFDAT_SNMP	Actions performed • Stops all active collections of the SNMP extension • Shuts down the SNMP extension master process
SHUTDOWN PERFDAT_EVA	Actions performed • Stops all active collections of the EVA extension • Shuts down the EVA extension master process

For detailed information about these commands please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

In order to start-up the VSI PERFDAT environment or parts of it one do not have to execute the LAUNCH / START commands of the PERFDAT_MGR utility. Common to all these start-up commands is that a sub-process is spawned and a command procedure is executed. Thus, the user can directly call these start-up command procedures from the DCL command prompt (Table 2.3).

Table 2.3 Startup script reference table

Command	Startup scripts executed
LAUNCH ALL	SYS$STARTUP:PERFDAT$STARTUP.COM SYS$STARTUP:PERFDAT_SNMP$STARTUP.COM SYS$STARTUP:PERFDAT_EVA$STARtUP.COM
LAUNCH PERFDAT	SYS$STARTUP:PERFDAT$STARTUP.COM
LAUNCH PERFDAT_SNMP	SYS$STARTUP:PERFDAT_SNMP$STARTUP.COM
LAUNCH PERFDAT_EVA	SYS$STARTUP:PERFDAT_EVA$STARTUP.COM
LAUNCH DQL$SRV	SYS$STARTUP:DQL$STARTUP.COM
LAUNCH PDBC$SRV	SYS$STARTUP:PDBC$STARTUP.COM
STARTNAME_SERVER	SYS$STARTUP:DQL_NAME$STARTUP.COM
START ARCHIVE	SYS$STARTUP:PERFDAT_ARCHIVE$STARTUP.COM

Almost all jobs of VSI PERFDAT have to run under the DQL$SRV user name and UIC[1]. Thus, if a user starts any of the VSI PERFDAT components using the PERFDAT_MGR LAUNCH commands the batch command script

SYS$STARTUP:PERFDAT$STARTUP_BATCH.COM

is executed. This batch command script submits the appropriatestartup scripts listed in table 2.3 into a batch queue on behalf of the DQL$SRV user. This startup batch queue can be user defined with the logical PERFDAT$STARTUP_QUEUE. If the batch queue referred by the logical exists and its status is idle, busy or available the startup scripts are submitted into this batch queue

[1]The DQL$SRV user account and is automatically created when VSI PERFDAT is installed on a system.

Otherwise the PERFDAT$STARTUP_BATCH.COM creates and initializes a temporary batch queue to execute the startup scripts.

The logical PERFDAT$STARTUP_QUEUE has to be defined system wide.

```
$ DEFINE/SYSTEM PERFDAT$STARTUP_QUEUE queue-name
```

In order to define the logical permanently it is strongly recommended to define the logical in:

SYS$STARTUP:PERFDAT$LOGICALS_CUSTOM.COM.

If this file does not exist in SYS$STARTUP copy the template file PERFDAT$CFG:PERFDAT$LOGICALS_CUSTOM.TEMPLATE either into SYS$COMMON:[SYS$STARTUP] or SYS$SPECIFIC:[SYS$STARTUP] depending if you want to maintain node-specifc logical definition files or you want to maintain just one common logical definition file which contains the node specific logicals.

Note

Starting with VSI PERFDAT V3.3 it is strongly recommended to use the startup command scripts rather than the PERFDAT_MGR LAUNCH commands. The VSI PERFDAT startup command scripts check the actual OpenVMS version in use before starting VSI PERFDAT. If OpenVMS has been upgraded all VSI PERFDAT version specific images are replaced and loaded automatically. The PERFDAT_MGR image is such a version specific image.

Performance data collection management

One can manually start and stop data collections of the OpenVMS data collector and the SNMP extension and monitor the status of the active collections on the local node. Table 2.4 summarizes the commands available for managing performance data collections.

Table 2.4 Command reference table for managing performance data collections

Command	Description
START COLLECTION *profile-name*	Start a new performance data collection according to the settings of the predefined collection profile *profile-name*. Qualifiers: /ADDRESS = *IP-address* This qualifier is mandatory if you start a remote data collection via the SNMP extension or the EVA extension (/OS_TYPE is not OpenVMS). You can enter the IP address of the remote system or the full qualified IP name. It is recommended to enter the IP address. /COMMUNITY This qualifier is optional if you start a remote data collection via the SNMP extension or the EVA extension (/OS_TYPE is not OpenVMS). Enter the community name to be used for SNMP *GET* requests.Since the SNMP community string check is case sensitive it is strongly recommended to use quotation marks when you specify the community string. /DEVICE = *EVA access device (1GGAxxx)* The /DEVICE qualifier is also mandatory if you start a performance data collection for a HP StorageWorks Virtual Array (EVA). This qualifier defines the access device to the EVA system. /NODE = *node name* This qualifier is mandatory if you start a remote data collection via the SNMP or EVA extension (/OS_TYPE is not OpenVMS). Enter the node name of the remote system. /FLUSH_TIME = *time of day* Each performance data collection started creates a new data file daily. With the /FLUSH_TIME qualifier you can define the time of day the new data file shall be created. If you omit the qualifier new data files are created at day change. /OS_TYPE = *keyword* Defines the system or applicationthe profile *profile-name* is valid for. If you want to start an OpenVMS data collection the qualifier can be omitted. If you want to start a data collection for a non-OpenVMS system (either via the SNMP, the EVA extension or by use of the VSI PERFDAT API) the qualifier is mandatory. Supported keywords are • OpenVMS • Tru64 • Brocade

- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

/OPENVMS_STYLE

The optional /OPENVMS_STYLE qualifier is only valid for EVA (HP StorageWorks Enterprise Array) data collections. It defines whether the performance data of a virtual disk with an OS unit ID assigned that is greater than zero will be stored using the OpenVMS FC device format (1DGAxxx, where xxx = OS unit ID of the virtual disk) or with its friendly name assigned by CV/EVA (CommandView/EVA).

/SHARE

If you start a data collection with the /SHARE qualifier the data of the actual collection is online accessible via the DQL interface.

The penalty is that the overall system performance may suffer due to excessive locking activity.

/SOURCE_ADDRESS

This qualifier is only valid for SNMP performance data collections. It defines the source IP address of the UDP/IP socket to be created and used by the SNMP extension to request SNMP performance data from a particular non OpenVMS system.

STOP COLLECTION *profile-name*

Stops an active performance data collection started with the collection profile *profile-name*.

Qualifiers:

/NODE = *node name*

This optional qualifier can be applied if one wants to stop an active performance data collection for a specific node.

/OS_TYPE = keyword

Defines the system or applicationthe profile *profile-name* is valid for. If you want to stop an OpenVMS data collection the qualifier can be omitted. If you want to stop a data collection for a non-OpenVMS system the qualifier is mandatory. Supported keywords are

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

If you enter no additional qualifier all active collections for all nodes that match the *profile-name* parameter and the

	operating system type defined by this qualifier (if the qualifier is omitted OpenVMS is assumed) are stopped.
SUBMIT COLLECTION *profile-name*	Schedules a new performance data collection to be started using the collection profile *profile-name*. In contrast to the START COLLECTION command the data collection is not started directly, but the command is forwarded to the scheduler of VSI PERFDAT. This command is only valid for submitting OpenVMS collections The OpenVMS data collector checks if any active collection is using the same profile name. In that case the start command is rejected. Depending on the qualifiers applied, the performance data collection is periodically retriggered or a single shot collection. Qualifiers: /AFTER = OpenVMS date & time format Defines the time the scheduler triggers the performance data collection the first time. If you omit the qualifier the collection is started immediately. /UNTIL = OpenVMS date & time format Defines the time the scheduler stops the collection. If you omit the qualifier the collection stays active until you stop it manually. /RESTART = OpenVMS delta date & time format With the /RESTART_INTERVAL qualifier you can define cyclic performance data collection activations. One the performance collection is started it will be periodically re-activated with a time delay defined by this qualifier. If you omit the qualifier the scheduled collection is a single shot collection. This qualifier requires the /AFTER and /UNTIL qualifier to be defined too. /SHARE If you start a performance data collection with the /SHARE qualifier the data of the actual collection is online accessible via the DQL interface. The penalty is that the overall system performance may suffer due to excessive locking activity.
SHOW COLLECTION *profile-name*	Shows the status of an active performance data collection started with the collection profile *profile-name*. VSI PERFDAT V3.0 and higher versions provide full wildcard support for the *profile-name* parameter. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within the string. Qualifiers: /ADVANCED Displays advanced information of the active collections. /BRIEF

	Displays summary information of all collection active – OpenVMS data collector, the SNMP extension, the EVA extension and application data collections. /NODE=*node_name* Applying the /NODE qualifier displays all the performance collections active on/for the node defined by its value. VSI PERFDAT V3.0 and higher versions provide full wildcard support for the *node-name* string. /OS_TYPE = keyword The /OS_TYPE qualifiers can be applied to selectively display the status of active performance data collections for systems or applications specified by this qualifier. Supported keywords are • OpenVMS • Tru64 • Brocade • EVA • Solaris • Linux • Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

All commands listed in Table 2.4 except the SUBMIT COLLECTION command are valid for managing data collections of the OpenVMS data collector as well as of the SNMP and the EVA extension and application data collections. Which component PERFDAT_MGR forwards the request to (OpenVMS data collector, SNMP, EVA extension, VSI PERFDAT API) depends on the value assigned to the qualifier /OS_TYPE. If you omit any qualifier the request is sent to the OpenVMS data collector.

For detailed information about these commands please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

Managing data archiving

Table 2.5 summarizes the commands for managing the archiving process.

Table 2.5 Command reference table for managing the archiving process

Command	Description
START ARCHIVE	Starts up the auto archiving process.
STOP ARCHIVE	Shuts down of the auto archiving process.
DEFINE ARCHIVE	Changes the control parameters for the archiving process in the archive control table of the VSI PERFDAT configuration database. This is done by applying different qualifiers /ENABLE Enables archive processing /DISABLE Disable archive processing. /KEEP_DAYS = *integer number* Number of days to keep performance data collection files in the directory PERFDAT$DB_TREND. /TIME_OF_DAY = *OpenVMS time format* Defines the time the archiving process will be daily triggered. Any change of the parameters applied with the DEFINE command does not effect the active archiving process but its default settings used when the archiving process (re)starts.
SET ARCHIVE	The SET command changes the parameter of the volatile archive control table. Thus, this command dynamically changes the behaviour of the archiving process. The settings will be lost when restarting the archiving process. The parameters are changed by applying different qualifiers /ENABLE Enables archive processing. /DISABLE Disable archive processing. /KEEP_DAYS = *integer number* Number of days to keep performance data collection files in the directory PERFDAT$DB_TREND. /TIME = *OpenVMS date & time format* Next archive time (date & time).
SHOW ARCHIVE	Displays the parameter settings of the permanent and volatile archive control table.

For detailed information about these commands please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

Auto-start table

A user can add, modify, delete and view the entries in the auto-start table of the VSI PERFDAT configuration database. The auto-start table of the VSI PERFDAT configuration database contains the required start-up parameters for all nodes a collection shall be automatically started when launching the OpenVMS data collector, the VSI PERFDAT EVA extension, the VSI PERFDAT SNMP extension or when the VSI PERFDAT API is initialized. When the VSI PERFDAT environment is launched (re-launched) the OpenVMS data collector as well as the VSI PERFDAT EVA extension, and the VSI PERFDAT SNMP extension accesses this table. The VSI PERFDAT API accesses this table whenever it is initialized. All components check if any collections are defined to be started on the local node. This is done by checking the content of every entry in this table. (For additional information please refer to chapter VSI PERFDAT OpenVMS Data Collector and VSI PERFDAT SNMP extension).

The entries of that auto-start table are also read by the auto-trend engine to determine if any trend and capacity report shall be processed. It checks if the local node is registered in the auto-start table and/or if the local node is defined as an EVA or SNMP agent or it hosts an application data collection. If this is the case the reports defined in the report profile table of the VSI PERFDAT configuration database that can be applied are processed.

Table 2.6 summarizes the commands for managing the auto-start table

Table 2.6 Command reference table for managing the auto-start table

Command	Description
ADD AUTOSTART *node-name*	Register a new node (*node-name*) in the auto-start table by invoking the auto-start configuration wizard. Optional qualifier: /OS_TYPE Defines the system or application to be added. If it is an OpenVMS node the qualifier can be omitted. If the system is non-OpenVMS the qualifier is mandatory. Supported keywords are: • OpenVMS • Tru64 • Brocade • EVA • Solaris • Linux • Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database Depending on the /OS_TYPE qualifier the auto-start configuration wizard asks for different inputs.

DELETE AUTOSTART *node-name*

Deletes a node (*node-name*) from the auto-start table. Optional qualifier:

/OS_TYPE

Defines the system or application to be deleted. If it is an OpenVMS node the qualifier can be omitted. If the system is non-OpenVMS the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

MODIFY AUTOSTART *node-name*

Modifies the parameter in the auto-start table for an existing node (*node-name*) entry by invoking the auto-start configuration wizard. Optional qualifier:

/OS_TYPE

Defines the system or application to be modified. If it is an OpenVMS node the qualifier can be omitted. If the system is non-OpenVMS the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

Depending on the /OS_TYPE qualifier the auto-start configuration wizard asks for different inputs.

SHOW AUTOSTART *node-name*

Displays the entries of the auto-start database. The parameter *node-name* is optional. Optional qualifier:

/OS_TYPE

Defines the system or application to be displayed. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux

- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

Depending on the /OS_TYPE qualifier the auto-start configuration wizard prompts the user for different inputs.

/OS_TYPE = OpenVMS

- Collection profile to auto-start when launching the OpenVMS data collector. This field is also read by the auto-trend engine to determine the source collection database for capacity and trend report processing.
- Online access
- Auto report start time
  It defines the start date for data processing the next time the auto-trend engine is triggered (collected data are processed from this date/time on).
- Online alerting
  VSI PERFDAT V3.0 and higher versions provide a performance alerting (watchdog) feature. This feature provides real time monitoring of dedicated statistics of an active performance data collection. Whenever one of these statistics exceeds free definable thresholds for a definable period of time the system manager will be alerted via OPCOM messages and user definable command procedures.
- Alert definition file in case of online alerting is enabled.
  The alert definition file contains the alert blocks that define the statistics to monitor, the warning and critical threshold values, the file names of the user definable command procedures etc. The default alert definition file is

  PERFDAT$CFG:PERFDAT_ALERT_OPENVMS.CFG
- Data flush time
  Each performance data collection started creates a new data file daily. The time entered at the data flush time prompt defines the time of day a new data file shall be created for the auto-started performance data collection. Enter a valid time string only.

/OS_TYPE = (Tru64, Brocade, Solaris, Linux)

- Collection profile to auto-start when launching the SNMP extension. This field is also read by the auto-trend engine to determine the source collection database for capacity and trend report processing.
- IP address of the remote node
- SNMP community string
- Agent node
  The agent node defines where to run the SNMP data collection and the node to run the auto-trend engine for processing data collected for the remote system referred by this entry.
- Source IP address to use
  If the SNMP agent (OpenVMS node that run the SNMP data collection) has more than one IP address configured, you can define which source IP address will be used by the SNMP extension to request the SNMP performance data from the target system.
- Online access
- Auto report start time
  It defines the start date for data processing the next time the auto-trend engine is triggered (collected data are processed from this date/time on).
- Online alerting
  First introduced with V3.0, PERFDAT provides a performance alerting (watchdog) feature for real time monitoring of dedicated statistics collected by an active performance data collection. Whenever one of these statistics exceeds free definable thresholds for a definable period of time the system manager will be alerted via OPCOM messages and user definable command procedures.
- Alert definition file in case of online alerting is enabled.
  The alert definition file contains the alert blocks that define the statistics to monitor, the warning and critical threshold values, the file names of the user definable command procedures etc. Depending on the value of the /OS_TYPE qualifier the default alert definition file is:
  - TRU64
    PERFDAT$CFG:PERFDAT_ALERT_TRU64.CFG
  - BROCADE
    PERFDAT$CFG:PERFDAT_ALERT_BROCADE.CFG
  - No default alert definition files are available for SOLARIS and LINUX
- Data flush time
  Each performance data collection started creates a new data file daily. The time entered at the data flush time prompt defines the time of day a new data file shall be created for the auto-started performance data collection. Enter a valid time string only.

/OS_TYPE = EVA

- Collection profile to auto-start when launching the EVA extension. This field is also read by the auto-trend engine to determine the source collection database for capacity and trend report processing.
- EVA access device

Enter the console access device to the EVA (HP StorageWorks Virtual Array) system you want to monitor. You can access the console of an EVA system only if the 'Console LUN ID' parameter of the EVA system is greater than zero. If the 'Console LUN ID' parameter is greater than zero, and you have executed the MCR SYSMAN IO AUTOCONFIGURE command you will get a 1GGAxxx device, where xxx = 'Console LUN ID' parameter value of the EVA system. This is the device you have to enter.

- Agent node
  The agent node defines where to run the EVA data collection and the node to run the auto-trend engine for processing data collected for the EVA system referred by this entry.
- Online access
- Auto report start time
  It defines the start date for data processing the next time the auto-trend engine is triggered (collected data are processed from this date/time on).
- Online alerting
  First introduced with V3.0, PERFDAT provides a performance alerting (watchdog) feature for real time monitoring of dedicated statistics collected by an active performance data collection. Whenever one of these statistics exceeds free definable thresholds for a definable period of time the system manager will be alerted via OPCOM messages and user definable command procedures.
- Alert definition file in case of online alerting is enabled.
  The alert definition file contains the alert blocks that define the statistics to monitor, the warning and critical threshold values, the file names of the user definable command procedures etc. The default alert definition file is:
  - PERFDAT$CFG:PERFDAT_ALERT_EVA.CFG
- Data flush time
  Each performance data collection started creates a new data file daily. The time entered at the data flush time prompt defines the time of day a new data file shall be created for the auto-started performance data collection. Enter a valid time string only.

/OS_TYPE = *application-name*

- Collection profile to be used by the VSI PERFDAT API to auto-start an application data collection when a process of the application defined by the /OS_TYPE qualifier (*application-name* parameter) is started on the node defined by the auto-start *node_name* parameter of the ADD AUTOSTART command.
- Online access
- Auto report start time
  It defines the start date for data processing the next time the auto-trend engine is triggered (collected data are processed from this date/time on).
- Online alerting
  First introduced with V3.0, PERFDATprovides a performance alerting (watchdog) feature for real time monitoring of dedicated statistics

collected by an active performance data collection. Whenever one of these statistics exceeds free definable thresholds for a definable period of time the system manager will be alerted via OPCOM messages and user definable command procedures.

- Alert definition file in case of online alerting is enabled.
  The alert definition file contains the alert blocks that define the statistics to monitor, the warning and critical threshold values, the file names of the user definable command procedures etc.

The predefined alert definition files

- PERFDAT$CFG:PERFDAT_ALERT_OPENVMS.CFG
- PERFDAT$CFG:PERFDAT_ALERT_TRU64.CFG
- PERFDAT$CFG:PERFDAT_ALERT_BROCADE.CFG
- PERFDAT$CFG:PERFDAT_ALERT_EVA.CFG

are part of the distribution kit.

For detailed information about these commands, the auto-start configuration wizard and how to configure alert blocks within an alert definition file please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

Archive control table

Table 2.7 summarizes the commands for managing the archive control table of the VSI PERFDAT configuration database.

Table 2.7 Command reference table for managing the archive control table

Command	Description
DEFINE ARCHIVE	Changes the control parameters for the archiving process in the archive control table of the VSI PERFDAT configuration database. This is done by applying different qualifiers /ENABLE Enables archive processing /DISABLE Disable archive processing. /KEEP_DAYS = *integer number* Number of days to keep performance data collection files in the directory PERFDAT$DB_TREND. /TIME_OF_DAY = *OpenVMS time format* Defines the time the archiving process will be daily triggered. Any change of the parameters applied with the DEFINE command does not effect the active archiving process but its default settings used when the archiving process (re)starts.

For detailed information about these commands please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

Collection profile table

Any performance data collection is profile controlled. These collection profiles are stored in the collection profile table of the VSI PERFDAT configuration database. A user can add, copy, modify, delete, import, export and view the collection profiles defined in that table. Table 2.8 summarizes the commands for managing the collection profile table

Table 2.8 Command reference table for managing the collection profile table

Command	Description
ADD PROFILE *profile-name*	Adds a new collection profile named *profile-name* to the collection profile table by invoking the collection profile configuration wizard. Optional qualifier: /ADVANCED Depending on the /OS_TYPE qualifier the profile configuration wizard provides additional profile configuration options. For more detailed information about the /ADVANCED qualifier please refer to the PERFDAT_MGR Reference Section of the manual VSI PERFDAT – PERFDAT_MGR Refernce manual. /OS_TYPE Defines the system or applicationthe collection profile is valid for. If the collection profile is valid for OpenVMS the qualifier can be omitted. If the collection profile is valid for non-OpenVMS the qualifier is mandatory. Supported keywords are: • OpenVMS • Tru64 • Brocade • EVA • Solaris • Linux • Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database Depending on the /OS_TYPE qualifier the collection profile wizard asks for different inputs.
COPY PROFILE *old-profile new-profile*	Copies the existing collection profile *old-profile* to the new collection profile *new-profile*. Optional qualifier: /OS_TYPE Defines the system or applicationthe collection profile is valid for. If the collection profile is valid for OpenVMS the qualifier can be omitted. If the collection profile is valid for non-OpenVMS the qualifier is mandatory. Supported keywords are: • OpenVMS • Tru64 • Brocade

- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

DELETE PROFILE *profile-name*

Deletes the existing collection profile named *profile-name* from the collection profile table. Optional qualifier:

/OS_TYPE

Defines the system or applicationthe collection profile is valid for. If the collection profile is valid for OpenVMS the qualifier can be omitted. If the collection profile is valid for non-OpenVMS the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

EXPORT PROFILE *profile-name*

Exports an existing collection profile defined by the *profile_name* parameter from the collection profile table of the VSI PERFDAT configuration database to a transport file.

Mandatory Qualifier:

/FILENAME

File name of the transport file.

Optional qualifiers:

/OS_TYPE

Defines the system or applicationthe exported collection profile is valid for. If the collection profile is valid for OpenVMS the qualifier can be omitted. If the collection profile is valid for non-OpenVMS the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

IMPORT PROFILE *profile-name*

Imports a collection profile defined by the *profile_name* parameter from a transport file to the collection profile table of the VSI PERFDAT configuration database.

Mandatory Qualifier:

/FILENAME

File name of the transport file.

Optional qualifiers:

/OS_TYPE

Defines the system or applicationthe imported collection profile is valid for. If the collection profile is valid for OpenVMS the qualifier can be omitted. If the collection profile is valid for non-OpenVMS the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

/UPDATE

If the collection profile already exists in the collection profile table of the VSI PERFDAT configuration database you can apply the /UPDATE qualifier to update that collection profile.

MODIFY PROFILE *profile-name*

Modifies the collection profile *profile-name* by invoking the collection profile configuration wizard. Optional qualifier:

/ADVANCED

Depending on the /OS_TYPE qualifier the profile configuration wizard provides additional profile configuration options. For more detailed information about the /ADVANCED qualifier please refer to the PERFDAT_MGR Reference Sectionof the manual VSI PERFDAT – PERFDAT_MGR Refernce manual.

/OS_TYPE

Defines the system or applicationthe collection profile is valid for. If the collection profile is valid for OpenVMS the qualifier can be omitted. If the collection profile is valid for non-OpenVMS the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux

	• Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database
SHOW PROFILE *profile-name*	Displays the collection profiles configured in the collection profile table. If *profile-name* is omitted all collection profiles are displayed. VSI PERFDAT V3.0 and higher versions provide full wildcard support for the *profile-name* parameter. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within the string.Optional qualifiers: /ADVANCED Depending on the /OS_TYPE qualifier advanced profile configuration options are displayed. For more detailed information about the /ADVANCED qualifier please refer to PERFDAT_MGR Reference Sectionof the manual VSI PERFDAT – PERFDAT_MGR Refernce manual. /BRIEF Displays summary information of all collection profiles configured in the collection profile table. /OS_TYPE Displays the collection profiles valid for the system or application specified by the /OS_TYPE qualifier. Supported keywords are: • OpenVMS • Tru64 • Brocade • EVA • Solaris • Linux • Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

The collection profile wizard prompts the user for the sample interval and the metrics to enable. Since the metrics available for the supported systems differ, the profile collection wizard prompts for different inputs, depending on the /OS_TYPE qualifier. The statistics included in each metric is listed in Appendix A.

For detailed information about these commands and the collection profile wizard please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

License table

Table 2.9 summarizes the commands for managing the license table of the VSI PERFDAT configuration database.

Table 2.9 Command reference table for managing the license table

Command	Description
CHECK LICENSE	Reads the license table and displays the status of each license key found (type of license, valid/expired).
LOAD LICENSE *key*	Checks if the license *key* entered is valid and loads the *key* into the license table.
UNLOAD LICENSE *key*	Deletes the license *key*.

For detailed information about these commands please refer to the manual VSI PERFDAT– PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

Record descriptor table

Table 2.10 shows the command for managing the record descriptor table of the VSI PERFDAT configuration database.

Table 2.10 Command reference table for managing the record descriptor table

Command	Description
LOAD METRIX *filename*	Loads metric and record descriptors from a valid *filename* into the record descriptor table of the VSI PERFDAT configuration database.

For detailed information about these commands please refer to the manual VSI PERFDAT – PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

This command is reserved for use by VSI support only.

Report profile table

Trend, capacity and baseline reports are extracted from performance data either via the auto-trend engine or manually via DQL$. In either case these reports are profile controlled. These report profiles are stored in the report profile table of the VSI PERFDAT configuration database. The user can add, copy, modify, delete, import, export and view the report profiles defined in that table. Table 2.11 summarizes the commands for managing the collection profile table.

Table 2.11 Command reference table for managing the collection profile table

Command	Description
ADD REPORT *report-name*	Adds a new report profile named *report-name* to the report profile table by invoking the report profile configuration wizard. Optional qualifier: /OS_TYPE Defines the system or application the report profile is valid for. If the report profile is valid for OpenVMS the qualifier can be omitted. If the report profile is valid for non-OpenVMS, the qualifier is mandatory. Supported keywords are: • OpenVMS • Tru64 • Brocade • RDB • EVA • Solaris • Linux • CACHE • Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database
COPY REPORT *old-report new-report*	Copies an existing report profile *old-report* to the new report profile *new-report*. Optional qualifier: /OS_TYPE Defines the system or application the report profile is valid for. If the report profile is valid for OpenVMS the qualifier can be omitted. If the report profile is valid for non-OpenVMS, the qualifier is mandatory. Supported keywords are: • OpenVMS • Tru64 • Brocade • RDB • EVA • Solaris • Linux

- CACHE
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

DELETE REPORT *report-name*

Deletes the existing report profile *report-name* from the report profile table. Optional qualifier:

/OS_TYPE

Defines the system or application the report profile is valid for. If the report profile is valid for OpenVMS the qualifier can be omitted. If the report profile is valid for non-OpenVMS, the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- RDB
- EVA
- Solaris
- Linux
- CACHE
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

EXPORT REPORT *report-name*

Exports an existing report profile defined by the *report_name* parameter from the report profile table of the PERFDAT configuration database to a transport file.

Mandatory Qualifier:

/FILENAME

File name of the transport file.

Optional qualifiers:

/OS_TYPE

Defines the system or application the exported report profile is valid for. If the report profile is valid for OpenVMS the qualifier can be omitted. If the report profile is valid for non-OpenVMS the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- RDB
- EVA
- Solaris
- Linux
- CACHE

- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

IMPORT REPORT *report-name*

Imports a report profile defined by the *report_name* parameter from a transport file to the report profile table of the PERFDAT configuration database.

Mandatory Qualifier:

/FILENAME

File name of the transport file.

Optional qualifiers:

/OS_TYPE

Defines the system or applicationthe imported report profile is valid for. If the report profile is valid for OpenVMS the qualifier can be omitted. If the report profile is valid for non-OpenVMS the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade
- RDB
- EVA
- Solaris
- Linux
- CACHE
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

/SOURCE

This qualifier can be applied to select a different source collection profile as that stored in the report header section if an existing report profile is updated with the imported report profile.

/UPDATE

If the report profile already exists in the report profile table of the PERFDAT configuration database you can apply the /UPDATE qualifier to update the layout and statistics section of that report profile.

MODIFY REPORT *report-name*

Modifies the report profile *report-name* by invoking the report profile configuration wizard. Optional qualifier:

/OS_TYPE

Defines the system or application the report profile is valid for. If the report profile is valid for OpenVMS the qualifier can be omitted. If the report profile is valid for non-OpenVMS, the qualifier is mandatory. Supported keywords are:

- OpenVMS
- Tru64
- Brocade

	• RDB • EVA • Solaris • Linux • CACHE • Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database
SHOW REPORT *report-name*	Displays the collection profiles configured in the report profile table. If *report-name* is omitted all report profiles are displayed. PERFDAT V3.0 and higher versions provide full wildcard support for the *report-name* parameter. Asterisk (*) and percent sign (%) wildcard characters can be placed anywhere within the string.Optional qualifiers: /BRIEF Displays summary information of all report profiles configured in the collection profile table. /OS_TYPE Displays only the report profiles valid for the system or application specified by the /OS_TYPE qualifier. Supported keywords are: • OpenVMS • Tru64 • Brocade • RDB • EVA • Solaris • Linux • CACHE • Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

The ADD REPORT and MODIFY REPORT command invoke the report profile wizard to add or modify a trend or capacity report profiles.

The report profile wizard consists of two sections

- Report header section (Trend, capacity and baseline report)
- Report data file layout and statistics section

For detailed information about these commands and the report profile wizard please refer to the manual VSI PERFDAT – PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

Online performance alert management

Online performance alerting can be dynamically enabled and disabled for any an active performance data executed by the OpenVMS data collector, the SNMP extension, the EVA extension or the VSI PERFDAT API.Table 1.11 summarizes the commands available for managing online performance alerting.

Table 1.11 Command reference table for managing online performance alerting

Command	Description
DISABLE ALERT *collection-profile*	Disable online alerting. The *collection-profile* name parameter specifies the active performance data collection that has online alerting enabled. With the SHOW COLLECTION command you can check if online alerting is enabled for the performance data collection specified by the *collection-profile* parameter. Qualifiers: /NODE = *node name* This qualifier is mandatory if you want to disable online alerting for an active non-OpenVMS performance collection. It specifies the node name of the remote system to disable online alerting /OS_TYPE = *keyword* The /OS_TYPE qualifier defines the system or application that runs a data collection started with the collection profile defined by the *collection-profile* parameter.In order to disable online alerting for an active non-OpenVMS performance data collection the /OS_TYPE qualifier is mandatory. If you disable online alerting for an active OpenVMS performance data collection profile you can omit the qualifier since OpenVMS is the default. Supported keywords are • OpenVMS • Tru64 • Brocade • EVA • Solaris • Linux • Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database
ENABLE ALERT *collection-profile*	Enable online alerting. The *collection-profile* name parameter specifies the active performance collection that has online alerting disabled. With the SHOW COLLECTION command you can check if online alerting is disabled for the performance collection specified by the *collection-profile* parameter. Qualifiers: /ALERT_FILENAME = *filename* The /ALERT_FILENAME qualifier specifies the alert definition file to use.

If you omit the /ALERT_FILENAME qualifier default alert definition files are used depending on the value of the /OS_TYPE qualifier as listed below. The files are located in PERFDAT$CFG:

OS_TYPE	Default alert definition file
OpenVMS	PERFDAT_ALERT_OPENVMS.CFG
Tru64	PERFDAT_ALERT_TRU64.CFG
Brocade	PERFDAT_ALERT_BROCADE.CFG
EVA	PERFDAT_ALERT_EVA.CFG

No default alert definition files are available for Solaris and Linux systems and any application. Thus, if you omit the /ALERT_FILENAME qualifier for Solaris, Linux or any application data collections the command fails.

/NODE = *node name*

This qualifier is mandatory if you want to enable online alerting for an active non-OpenVMS performance collection. It specifies the node name of the remote system to disable online alerting.

/OS_TYPE = *keyword*

The /OS_TYPE qualifier defines the system or application that runs a data collection started with the collection profile defined by the *collection-profile* parameter.In order to enable online alerting for an active non-OpenVMS performance collection the /OS_TYPE qualifier is mandatory. If you enable online alerting for an active OpenVMS performance data collection you can omit the qualifier since OpenVMS is the default. Supported keywords are

- OpenVMS
- Tru64
- Brocade
- EVA
- Solaris
- Linux
- Name of any application that uses the VSI PERFDAT API to insert data into the VSI PERFDAT performance database

CHECK ALERT *alert-definition-file*	This command reads the alert definition file defined by the *alert-definition-file* parameter and checks if all alert blocks defined within are valid. If an invalid line item is detected the line item and the line number is displayed.

For detailed information about these commands and how to define alert blocks please refer to the manual VSI PERFDAT – PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

Performance database file name cache service management

The distributed performance database is organized in way that there exists no persistent root file for any VSI PERFDAT performance database on disk (see

chapter VSI PERFDAT distributed performance database) All meta-data (field and record descriptors, data link descriptors, index reference table descriptor …) necessary to access the data is stored in the header of each physical storage area. The advantage is that data files can be moved to any OpenVMS node and the data file stays read accessible without any additional actions such as data conversion, unload and load operations. On the other hand the meta-data have to (re)fetched any time a user connects to the distributed performance database via the DQL$ utility or the GUI in order to create a virtual root file (data link cache) required to access the data.

Prior to VSI PERFDAT V3.2 the meta-data were fetched by performing a full data file header scan on all members of the PERFDAT community. Thus, prior to VSI PERFDAT V3.2 the initial connect request to the distributed performance database took few seconds up to minutes.

In order to solve that performance issue VSI PERFDAT provides the performance database file name cache service DQL_NAME first introduced with VSI PERFDAT V3.2.

The performance database file name cache service DQL_NAME provides a database file name cache to all VSI PERFDAT components that contains full header information about all VSI PERFDAT database files locally stored. As long as the performance database file name cache service DQL_NAME is available and the database file name cache is marked valid all VSI PERFDAT components obtain database file header information from that cache rather than scanning the files on disk. Thus, the initial connect request speeds up dramatically (ten times and more) compared to VSI PERFDAT V3.1 and lower versions of VSI PERFDAT.

Table 1.12 summarizes the commands for managing the performance database file name cache service DQL_NAME.

Table 1.12 Command reference table for managing the performance database file name cache service DQL_NAME

Command	Description
START NAME_SERVER	Starts up the performance database file name cache service DQL_NAME.
STOP NAME_SERVER	Shuts down of the performance database file name cache service DQL_NAME.
FLUSH NAME_SERVER	Flushes the performance database file name cache on the local node and triggers the DQL_NAME service to rebuild it.
SET NAME_SERVER	This command is used to change the time to leave duration of the performance database file name cache entries.
	/TTL
	The /TTL qualifier defines the new cache time to leave duration in minutes. The /TTL qualifier is mandatory. The new TTL value takes effect immediately after the current TTL period has expired.
SHOW NAME_SERVER	Displays the current TTL setting of the performance database file name cache service DQL_NAME.

For detailed information about these commands please refer to the manual VSI PERFDAT – PERFDAT_MGR Reference Manual or the online help of PERFDAT_MGR.

VSI PERFDAT Directory structure and Logicals

This chapter provides a short description of the directories and VSI PERFDAT logicals. This logicals control the overall behaviour of the VSI PERFDAT environment. Most of the logicals are automatically defined by the installation procedure. However there are some logicals that are subject to change in order to manipulate the behaviour of some components of VSI PERFDAT.

VSI PERFDAT Directory structure

- PERFDAT$COMMON
  This concealed device is the root directory of the common VSI PERFDAT directory tree.

- PERFDAT$SPECIFIC
  This concealed device is the root directory of the local VSI PERFDAT database directory (PERFDAT$SPECIFIC:[DB])which contains the data files actually accessed by the data collectors (OpenVMS data collector, EVA extension& SNMP extension).

- PERFDAT$ALERT
  The directory referred by this logical contains the alert log files created by the performance online alerting subsystem, and is used as the temporary storage location for alert content files.

- PERFDAT$BIN
  The directory referred by this logical contains all VSI PERFDAT images and COM files.

- PERFDAT$COMMON:[BIN.AXP.V722]
  This directory contains all OpenVMS V7.2-2 Alpha-specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.AXP.V73]
  This directory contains all OpenVMS V7.3 Alpha-specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.AXP.V731]
  This directory contains all OpenVMS V7.3-1 Alpha-specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.AXP.V732]
  This directory contains all OpenVMS V7.3-2 Alpha-specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.AXP.V82]
  This directory contains all OpenVMS V8.2 Alpha-specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.AXP.V83]
  This directory contains all OpenVMS V8.3 Alpha-specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.IA64.V82]
  This directory contains all OpenVMS V8.2 I64 specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.IA64.V821]
  This directory contains all OpenVMS V8.2-1 I64 specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.IA64.V83]
  This directory contains all OpenVMS V8.3 I64 specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.IA64.V831]
  This directory contains all OpenVMS V8.3-1H1 I64 specific VSI PERFDAT images.

- PERFDAT$COMMON:[BIN.IA64.V84]
  This directory contains the VSI PERFDAT images for:
  - OpenVMS V8.4 I64
  - OpenVMS V8.4-1H1 I64
  - OpenVMS V8.4-2 I64
  - OpenVMS V8.4-2L1 I64

- PERFDAT$CFG
  The directory referred by this logical contains all configuration databases (VSI PERFDAT configuration database, CSV mapping database …) default alert definition files and template CFG files.

- PERFDAT$DB
  This logical is a directory search list. All the directories referred by PERFDAT$DB are searched for VSI PERFDAT data files on a DQL access.
    - PERFDAT$DB_LOCAL
      Contains the data files actually accessed by the data collectors (OpenVMS data collector, EVA extension& SNMP extension)
    - PERFDAT$DB_ARCHIVE
      On an archive node this logical refers to the common data repository. On any other node the logical refers to local data repository and data buffering directory in the case the archive node is not reachable for any reason the time the archiving process is triggered. PERFDAT$DB_ARCHIVE is the working directory of the archiving process (all data that is older as KEEP_TIME are deleted).
    - PERFDAT$DB_SAVE
      Save directory for base line data. That directory is not accessed by the archiving process.
    - PERFDAT$DB_TREND
      Contains the trend and capacity report data files created by the auto trend engine or the DQL$ utility.
- PERFDAT$EXAMPLES
  This directory contains C programming examples that illustrate how to use the VSI PERFDAT API.
- PERFDAT$GRAPH
  Default storage location for graphs (PNG format) created with the CREATE GRAPH command of the DQL$ utility.
- PERFDAT$HELP
  This directory contains the help library files for the PERFDAT_MGR and DQL$ utilities as well as VSI PERFDAT documentation ZIP file. All future VSI PERFDAT ECO release notes will be placed in this directory.
- PERFDAT$INCLUDE
  This directory contains the VSI PERFDAT API header files.
- PERFDAT$LIBRARY
  This directory contains the VSI PERFDAT API object libraries. These object libraries contain the VSI PERFDAT API callable routines:
    - PERFDAT$LIBRARY:PERFDAT_API_AXP.OLB
      Alpha object library
    - PERFDAT$LIBRARY:PERFDAT_API_IA64.OLB
      I64 object library
- PERFDAT$LOAD

Default storage location to import/load performance data created by different sources into the VSI PERFDAT distributed performance database.

- PERFDAT$LOG
  The directory referred by this logical contains all LOG files created by the components of the VSI PERFDAT environment.

- PERFDAT$STARTUP
  This directory contains the startup scripts for all components of VSI PERFDAT and all check command files required for multi version support.

- PERFDAT$SUPPORT
  This directory contains required software packages, documentation and readme files to be installed on non-OpenVMS systems files to enable VSI PERFDAT SNMP data collections. Currently this directory contains the recommended NET-SNMP packages for Solaris 2.6, 7, 8 and 9.

- PERFDAT$TOOLS
  The directory referred by this logical contains all VSI PERFDAT tools.

Control logicals

All VSI PERFDAT control logicals have to be defined system-wide.

VSI PERFDAT startup control logicals

- PERFDAT$STARTUP_QUEUE
  If a user starts up any of the VSI PERFDAT data collectors the appropriate startup command scripts are submitted into a batch queue for execution. If this logical exists and it refers to a valid batch queue the command scripts are submitted to this batch queue. A batch queue is considered valid if the queue exists and the status of the queue is IDLE, BUSY or AVAILABLE. For detailed information about this logical please refer to section VSI PERFDAT_MGR – Management Interface.

VSI PERFDAT common control logicals

- PERFDAT$OPCOM_CLASS
  This logical defines the operator classes the VSI PERFDAT components send their OPCOM messages to. Assign the operator classes as a comma separated list to this logical. If the logical is not defined or if all operator classes assigned to this logical are invalid the VSI PERFDAT components send their OPCOM messages to all operator classes.
  This logical becomes effective immediately whenever it is changed.

VSI PERFDAT OpenVMS data collector

- PERFDAT$DATA_MEANSIZE
  The mean record size of the data records stored in the collection data files (physical storage area) is defined by this logical. If not defined, the average record size is automatically calculated by the data collector depending on the profile definitions.

- PERFDAT$DATA_RECORDCNT
  It defines the maximum number of records that are expected to be written into a physical storage area. If not defined, the average record size is automatically calculated by the data collector depending on the profile definitions. By setting both logicals (DATA_MEANSIZE & DATA_RECORDCNT) you can define the initial file size of the data files created by the data collectors.

- PERFDAT$DO_NOT_INCREASE_SAMPLETIME
  At the end of the sample interval of each collection the OpenVMS data collector checks if all asynchronous database store operations (file writes) for the collection have been completed. If there are still some uncommitted transactions no data are sampled. Depending on the value of the logical different actions are performed:
  - If this logical is not defined or the value is FALSE the sample interval is doubled for the associated collection. This is done at maximum of 3 times. If the collection interval for a specific

collection interval has been doubled 3 times and there are still uncommitted inserts the collection will be stopped.
    - If the value of this logical is TRUE the sample interval is left unchanged. Be aware that in this case the OpenVMS data collector will not stop automatically although it might cause overall I/O performance problems.

- PERFDAT$DEBUG
  This logical defines the debug level of the VSI PERFDAT OpenVMS data collector. The value assigned to this system-wide logical defines the content of the debug output that is written into the log-files.
  Set this logical only if advised by VSI PERFDAT support.

- PERFDAT$FILE_CACHE_DEBUG
  This logical sets the VSI PERFDAT OpenVMS data collector to write debug output about the internal file-cacheinto its log-file. Set this logical onlyif advised by VSI PERFDAT support.

- PERFDAT$FILE_CACHE_TTL
  This logical defines the 'Time to Leave' parameter (file-cache TTL) of the internal file-cache of the VSI PERFDAT OpenVMS data collector in minutes. All file-cache entries that refer to files that have not been accessed during the file-cache TTL period are automatically removed from the file-cache. The default file-cache TTL is 15 minutes.

- PERFDAT$MAX_CPU_LOAD
  This logical defines the percentage of available CPU power that can be consumed by the OpenVMS data collector before dynamic resource trimming starts. If not defined, it is automatically set to 20% (default).

- PERFDAT$MAX_XFC_ALLOC
  Do not define the logical or modify the value assigned to except you are asked by VSI support.

- PERFDAT$MAX_FILE_CACHE_ALLOC
  This logical defines the maximum size of the internal file-cache of the VSI PERFDAT OpenVMS data collector in Bytes.

- PERFDAT$NO_INIT_FILE_CACHE
  During the initialization phase the VSI PERFDAT OpenVMS data collector fills its internal file cache. It fetches all file ID's known to the OpenVMS system (XFC & processes) and reads the corresponding file names from INDEXF.SYS. This may last a few seconds up to several minutes. During the initial file cache update the VSI PERFDAT OpenVMS data collector blocks any operator command.
  If this system-wide logical exists the initial file cache update is not performed.

- PERFDAT$NO_OPCOM
  If this system-wide logical exists, then the VSI PERFDAT OpenVMS data collector sends no OPCOM messages. This logical is checked

periodically. Thus, the logical can be changed without restarting the VSI PERFDAT OpenVMS data collector.

- PERFDAT$PMS_INITIAL_BUFFERS
  Do not define the logical or modify the value assigned to except you are asked by VSI support.

- PERFDAT$PMS_IO_EXPIRE_TIME
  Do not define the logical or modify the value assigned to except you are asked by VSI support.

- PERFDAT$PMS_MAXIMUM_BUFFERS
  Do not define the logical or modify the value assigned to except you are asked by VSI support.

VSI PERFDAT EVA extension

- PERFDAT_EVA$DEBUG
  This logical defines the debug level of the VSI PERFDAT EVA extension. The value assigned to this system-wide logical defines the content of the debug output that is written into the log-files of all components of the VSI PERFDAT EVA extension.
  Set this logical only if advised by VSI PERFDAT support.

- PERFDAT_EVA$NO_OPCOM
  If this system-wide logical exists, then the VSI PERFDAT EVA extension (master and working processes) sends no OPCOM messages. This logical is checked periodically. Thus, the logical can be changed without restarting the VSI PERFDAT EVA extension.

VSI PERFDAT SNMP extension

- PERFDAT_SNMP$DEBUG
  This logical defines the debug level of the VSI PERFDAT SNMP extension. The value assigned to this system-wide logical defines the content of the debug output that is written into the log-files of all components of the VSI PERFDAT SNMP extension.
  Set this logical only if advised by VSI PERFDAT support.

- PERFDAT_SNMP$MAX_RETRIGGER
  This logical defines the number of retries the VSI PERFDAT SNMP extension performs for a particular SNMP request if it got no answer with defined timeout period. You can assign any integer between 1 ... 30. If you enter a value greater then 30 the SNMP extension sets the retrigger count to the maximum value = 30. If the logical is omitted the default of 10 is used.

- PERFDAT_SNMP$PROBE_INTERVAL

The probe interval logical defines the SNMP timeout in seconds. This logical defines the time the VSI PERFDAT SNMP extension waits to receive an answer on a particular SNMP request. If no answer is received within the time period defined by this logical, the SNMP extension retriggers the request. You can assign any integer between 1 ... 60. If this logical is omitted the default of 2 seconds is used.

The maximum number of retries is defined by the logical PERFDAT_SNMP$MAX_RETRIGGER. If

*Retry count * probe interval > 60*

the probe interval is calculated according to

Probe interval = (int) (60 / retry count)

regardless of the value of this logical.

- PERFDAT_SNMP$NO_OPCOM
  If this system-wide logical exists, then the VSI PERFDAT SNMP extension (master and working processes) sends no OPCOM messages. This logical is checked periodically. Thus, the logical can be changed without restarting the VSI PERFDAT SNMP extension.

- PERFDAT_SNMP$SEQNR_OFFSET
  This logical defines the lowest sequence number to use for SNMP get request. Set this logical only if advised by VSI PERFDAT support.

VSI PERFDAT Archiving and Housekeeping

- PERFDAT$ARCHIVE_DAYS_TO_KEEP
  It defines the number of days that the archive shall keep collected performance raw data. This logical can be changed directly or via the PERFDAT_MGR command

  $ MCR PERFDAT_MGR SET ARCHIVE/KEEP_DAYS=x

- PERFDAT$ARCHIVE_ENABLED
  If defined as TRUE the automatic archiving is enabled. Valid values are TRUE/FALSE. It can be set directly or via the PERFDAT_MGR command

  $ MCR PERFDAT_MGR SET ARCHIVE/ENABLE

- PERFDAT$ARCHIVE_NODE
  This logical is pointing to your archive node and will be defined during installation. If you change the logical directly, please don't forget to update the entry in

  SYS$STARTUP:PERFDAT$LOGICALS_SPECIFIC.COM

- PERFDAT$NEXT_ARCHIVE_DATE
  It defines date/time for the next automatic archiving run. Logical can be set either directly or by the PERFDAT_MGR command

  $ MCR PERFDAT_MGR SET ARCHIVE/TIME=time

VSI PERFDAT mangement utility (PERFDAT_MGR)

- PERFDAT$ALLOW_UNAUTHORIZED_ACCESS
  Only privileged users are allowed to modify the VSI PERFDAT configuration database and performance collections. By setting this logical to TRUE you allow any user, who is able to access the VSI PERFDAT files (depends on file protection and process privileges) to modify configuration database and performance collections.

VSI PERFDAT query interface (DQL)

- PERFDAT$COMMUNITY
  It defines the data (node) community / view of the distributed performance database if that node is selected as the access server.

- PERFDAT$NODEDATA_HOSTED
  You can grant DQL access to collection databases or part of it stored on the local node that are not created by any member of the community the local node is member of. Assign the nodes that created these data files as a comma separated list to this logical.

- PERFDAT$NO_NODE_FILTER
  If set, DQL access to all data is permitted, even if the data was not created by the node or a member of the community.

- PERFDAT$SCRATCH
  The CREATE GRAPH command is used to create PNG formatted graphs that can be viewed directly with your WEB browser. This command facilitates automated WEB based graphing and data analysis. During command execution temporary CSV files are created. The default directory to store these temporary CSV files is PERFDAT$GRAPH. If the user who executes the CREATE GRAPH command is not owner of the PERFDAT$GRAPH directory audit alerts are triggered.
  To avoid such audit alerts the directory to store the temporary CSV files can be user defined. If the logical PERFDAT$SCRATCH exists and if it refers a valid directory all temporary CSV files created by the CREATE GRAPH command of the DQL$ utility are stored in this directory.

- DQL$CONNECT_RETRY
  If the connect request to the DQL$SRV service of a member of the PERFDAT community fails for any reason this logical defines the number of connect retries. The default is three.

- DQL$INQUIRY_TIMEOUT
  After the TCP/IP connection to the DQL$SRV of a member of the PERFDAT community has been established successfully the client (DQL$ utility or PDBC$SRV) requests the version of the DQL$SRV service. This logical defines the time in seconds the client waits for response of this version inquiry. If the client receives no response within the timeout

interval it retries the inquiry again according to the definition of the logical PERFDAT$CONNECT_RETRY.

- DQL$MAX_ELEMENT_CACHE_ENTRIES
  This logical defines the maximum number of elements cached by the DQL$ utility and PDBC$SRV server when performing a SHOW ELEMENT command using the ORDERED BY clause (= Sort request from the GUI). If the elements addressed by the SHOW ELEMENT query exceeds the value defined by this logical the sort query fails and the element list is not returned in order of the statistics defined by the ORDERED BY clause but in alphabetical order. The default value is 16384.

- DQL$MAX_DATA_CACHE_ENTRIES
  This logical defines the maximum number of data samples cached by the DQL$ utility and PDBC$SRV server when performing stacked data queries.If the data samples requested by a stacked data query exceeds the value defined by this logical the result table get corrupted. The default value is 24576. In case you request stacked data from a collection database with a sample interval of 120 sec. the 34 day is the maximum time period the stacked request can be applied to.

- DQL$MAX_DATA_TRANSFER
  This logical defines the maximum size of the data packets transferred between the DQL$ utility / PDBC$SRV server and the DQL$SRV service in kByte. The default value is 64. In case the TCP window size is smaller than the defaultdata transfer performance may speed up if you assigning a value to this logical that is smaller or equal the TCP window size.

- DQL$C_TCP_KEEPINIT
  This logical defines the timeout in seconds if the DQL$ utility and/or the PDBC$SRV server cannot establish a TCP connection to a DQL$SRV server process. If not defined the default timeout for initial connection request of the TCP/IP stack will be used. This logical can be used to speed up the initial connect and database query requests to the nodes of a community in case one of these nodes are not accessible or the DQL$SRV service is disabled.

- DQL$C_TCP_NODELAY
  This logical specifies if the components of the DQL query interface (DQL$SRV, PDBC$SRV and DQL$ utility) sends data via TCP to its partner when data is present even if outstanding data has not been acknowledged. The default value is FALSE. If you want to enable NODELAY assign TRUE to the logical.

- DQL$SRV_DEBUG
  This logical advises all DQL$SRV processes to write debug output into their log-files. Set this logical only if advised by VSI PERFDAT support.

- DQL$SRV_PORT

This system-wide logical defines the DQL$SRV service listener port. If this system-wide logical exists when the DQL$SRV starts up it creates the listener socket using the port number defined by this logical instead of the default port number 3879.

- PDBC$SRV_PORT
  This system-wide logical defines the PDBC$SRV service listener port. If this system-wide logical exists when the PDBC$SRV starts up it creates the listener socket using the port number defined by this logical instead of the default port number 5254.

VSI PERFDAT Auto-trend Engine

- PERFDAT$ARCHIVE_TRENDS
  If you assign TRUE to this system-wide logical the auto-trend engine automatically creates trend data files on the archive node if an archive node is configured, the archive node has VSI PERFDAT V3.3 or a higher version installed and the archive node is accessible. If one of these criteria is not fulfilled the auto-trend engine creates trend data files on the local node.

Tools & Utilities

This chapter provides a short description of the tools and utilities provided by VSI PERFDAT. All tools and utilities are located in the directory PERFDAT$TOOLS.

RDB performance data import utility

VSI PERFDAT provides the utility PERFDAT_IMPORT_RDB.EXE to import RDB performance data previously collected using the RMU/SHOW STATISTICS command.

General description

You can collect performance data for RDB using the RMU/SHOW STATISTICS command. By applying the /OUTPUT qualifier the RMU/SHOW STATISTICS command writes performance raw data (absolute counters) into a binary file. E.g. if you want to collect performance raw data for a RDB database from now until 1-May-2006 (assuming the time now is before 1-May-2006) with 60 sec. sample interval apply a command like:

```
$ RMU / SHOW STATISTICS                database_name –
                        /NOINTERCATIVE -
                        /OUTPUT=RDB_30-04-2006.STAT -
                        /UNTIL=”01-MAY-2006 00:00:00” -
                        /TIME = 60
```

For more information about collecting RDB performance data using the RMU/SHOW STATISTICS command please see the RMU online help or the Oracle RDB Guide to Database Performance and Tuning.

The VSI PERFDAT RDB performance data import utility PERFDAT_IMPORT_RDB reads the binary files created by the RMU/SHOW STATISTICS command, calculates meaningful statistics from the raw data stored in the binary file and imports these statistics into the VSI PERFDAT distributed performance database.

The VSI PERFDAT RDB performance data import utility is located in the directory PERFDAT$TOOLS:

PERFDAT$TOOLS:PERFDAT_IMPORT_RDB.EXE

Note

Once RDB performance data are imported to the VSI PERFDAT distributed performance database you can apply all methods and features (cluster views, stored procedures, correlation analysis, deviation analysis …) provided

by VSI PERFDAT to analyze RDB performance data. Each PERFDAT data file that contains RDB performance data also contains descriptions for all statistics inserted that are displayed by the GUI similar to statistics collected by the VSI PERFDAT OpenVMS data collector, the VSI PERFDAT EVA extension or the VSI PERFDAT SNMP extension.

To import RDB performance data into the PERFDAT distributed performance database create a foreign command symbol to the import utility:

```
$ RDB_IMPORT :== $PERFDAT$TOOLS:PERFDAT_IMPORT_RDB.EXE
```

and apply all input parameters.The utility accepts two input parameters:

- P1 Specifies the input directory of the binary RDB performance files. This parameter is mandatory. If you do not apply P1 the utility terminates immediately. The RDB performance data import utility processes all binary RMU/SHOW STATISTICS files that are located in that directory. This directory can contain binary input files created on the local node but also binary RMU/SHOW STATISTICS files created on other nodes. The import utility automatically checks the creator node of the files and inserts the data to the appropriate PERFDAT collection database (see explanation later on).
- P2 Sample interval of the target PERFDAT collection database to import the RDB performance data in seconds. This parameter is optional. If you omit parameter P2 the sample interval of the target PERFDAT performance database file to import the RDB performance data is equal to the sample interval in the binary RDB performance file. If P2 is greater than the sample interval of the binary RDB performance database file the statistics calculated from the raw data of the RDB files are averaged to the time boundaries defined by the P2 parameter. If P2 is less than the sample interval of the RMU/SHOW STATISTICS binary file the sample interval of the binary input file is used to import the RDB performance data.

Note

The utility processes all binary RDB performance data files stored in the directory specified by input parameter P1. It is recommended that the directory specified contains binary RMU/SHOW STATISTICS files only.

The RDB performance data import utility automatically creates the PERFDAT collection databases and physical storage areas to import the RDB performance data. The aliases of the PERFDAT collection databases created by this utility have the format:

*Nodename*_RDB-*sample_interval*SEC

The node name is fetched from the header of the binary RDB performance data file and the sample interval encoded in the PERFDAT database alias is the sample interval of the target PERFDAT collection database (either parameter P2

or the sample interval of the original binary file – see explanation of the input parameters). E.g. a binary RMU/SHOW STATISTICS file was created on node VMSTM1 with a sample interval of 60 seconds and input parameter P2 is omitted the RDB performance data will be imported to a PERFDAT collection database that is referenced by the alias:

VMSTM1_RDB-60SEC

If you import the RDB performance data of the same binary file with P2 present the data are imported to another PERFDAT collection database except P2 is equal to the sample interval of the binary RMU/SHOW STATISTICS file. E.g. P2 = 120 (sample interval of the target PERFDAT collection database in seconds) the RDB performance data will be imported to a PERFDAT collection database that is referenced by the alias:

VMSTM1_RDB-120SEC

Note

Thus, performance data of all RDB databases collected with the same sample interval on a node will be imported to the same PERFDAT collection database. If the sample interval of the binary RMU/SHOW STATISTICS files created on a node for different RDB databases is not equal and you want to have them imported to the same PERFDAT collection database apply parameter P2 when importing the RDB performance data. P2 has to be at least equal or greater then the greatest sample interval of all these RMU/SHOW STATISTICS binary files.

After a binary RMU/SHOW STATISTICS file has been import to the PERFDAT distributed performance database that file will be automatically copied to the directory PERFDAT$COMMON:[LOAD.PROCESSED] and deleted from the source directory. If the copy operation fails for any reason the user gets informed and the file will not be deleted from the source directory (parameter P1).

Note

All binary RMU/SHOW STATISTICS files that have been successfully processed and copied to PERFDAT$COMMON:[LOAD.PROCESSED] have to deleted from that directory by the user.

RDB metrics

All the RDB metrics available after importing binary RMU/SHOW STATISTICS filesusing the PERFDAT_IMPORT_RDB tool are listed below. A detailed description of the statistics of each metric is provided in Appendix A of this document.

- CACHE
- CACHE.UNMARK
- INDEX.HASH

- INDEX.INSERTION
- INDEX.REMOVAL
- INDEX.RETRIEVAL
- IO.ASYNCH_IO
- IO.FETCH
- IO.FILE
- IO.PREFETCH
- IO.STALL_IO
- JOURNAL.2PC
- JOURNAL.AIJ
- JOURNAL.ALS
- JOURNAL.DBR
- JOURNAL.RUJ
- LOCK.TYPE
- LOGNAM
- OBJECT.TYPE
- RECORD
- SNAPSHOT
- STALLS
- TRANS
- TRANS.HISTOGRAMM

Example

The following command sequence demonstrates how to import all the binary RMU/SHOW STATISTICS files located in the directory SYS$SYSDEVICE:[RDB] into the PERFDAT distributed database. The sample interval of the target PERFDAT collection databases shall be 120 seconds. If all binary RMU/SHOW STATISTICS files were collected with a sample interval less than 120 seconds applying P2 guarantees that the RDB performance data of all RDB databases collected on the same node are imported to the same PERFDAT collection database (see also the General description of the utility).

```
$ RDB_IMPORT :== $PERFDAT$TOOLS:PERFDAT_IMPORT_RDB
$
$ RDB_IMPORT “SYS$SYSDEVICE:[RDB]” “120”
```

After PERFDAT_IMPORT_RDB has processed all binary input files delete these files from the directory PERFDAT$COMMON:[LOAD.PROCESSED].

RDB trend and capacity reports

The RDB metrics listed in chapter RDB metrics are automatically imported to the VSI PERFDAT configuration database when VSI PERFDAT is installed. Thus, trend and capacity report profiles can be created for RDB data. Trend and capacity report for RDB are not automatically processed by the VSI PERFDAT auto-trend engine. Thus, having these reports automatically created you have to schedule a job on your own that executes the appropriate DQL$ EXTRACT command. For

information about creating trend and capacity reports using the DQL$ command EXTRACT please refer to the manual VSI PERFDAT – DQL$ Reference manual.

CACHE performance data import utility

The VSI PERFDAT utility PERFDAT$TOOLS:IMPORT_LOAD_CACHE.COM is designed to load or import CACHE (database of InterSystems Corporation) performance statistics collected by the MGSTAT utility of CACHE into the distributed VSI PERFDAT performance database.

General description

Three parameters can be passed to this tool:

- P1
  MGSTAT performance data file to load/ import
- P2
  Target database alias to import the content of the MGSTAT file
- P3
  If P3 contains the string DEBUG the CSV import file created to load the context of the MGSTAT utility and the DQL load/import script are not deleted.

The IMPORT_LOAD_CACHE.COM utility converts the MGSTAT file provided in P1 into a CSV file formatted as required by the DQL$ LOAD/IMPORT commands. It creates a DQL load script (*.DQL) and executes this DQL load script automatically to load/import the context of the MGSTAT file into the distributed VSI PERFDAT performance database.

If P2 is omitted the utility performs a load operation. In this case the target VSI PERFDAT collection database alias is automatically defined:

*Nodename*_CACHE-*sample-interval*SEC

The *nodename* and the *sample-interval* parameters are automatically extracted from the header of the MGSTAT file.

When the utility loads MGSTAT data into the target VSI PERFDAT collection database the database files associated with the target VSI PERFDAT collection database are automatically created if they do not already exist.

If P2 contains a VSI PERFDAT database alias the MGSTAT data is imported into the collection database defined by P2. When data is imported into an existing collection database the data is normalized. This means that based on the MGSTAT data, expectancy values are calculated for the timestamps stored in the target collection database, and these expectancy values are inserted into the collection database. Normalizing data into an existing collection database is required to guarantee that all statistical methods provided by the DQL interface can also be applied to this imported data.

When data is loaded into a collection database no such pre-processing is performed. Thus, it is not guaranteed that all statistical methods provided by the DQL interface can be applied to the loaded data except that all MGSTAT data files contain the same time series (same time-stamps).

One MGSTAT file contains the performance data of one CACHE database instance running on a particular node. Thus, if the MGSTAT files are from a

particular time period (MGSTAT data of different CACHE instances and nodes) will be loaded into one collection database proceed as described below:

1. Load one MGSTAT data file into the distributed VSI PERFDAT collection database by omitting P2
2. Import all other MGSTAT files into the collection database created by passing the collection database alias of the collection database created when the first MGSTAT file was loaded in P2.

For detailed information about loading and importing data into the distributed VSI PERFDAT performance database please refer to the online help of the DQL$ utility or the manual VSI PERFDAT – DQL$ Reference manual.

Generic CSV load utility

PERFDAT_LOADCSV.COM located in PERFDAT$TOOLS is a generic CSV load utility to load a bunch of CSV files containing data of any kind into the VSI PERFDAT distributed performance database at once.

Prerequisite to load data from CSV files into the VSI PERFDAT distributed database using this tool is that the format of the CSV files is supported by DQL. For detailed information about supported CSV formats please refer to the MAP command description in the manual VSI PERFDAT – DQL$ Reference Manual.

Five input parameters can be applied when calling the utility:

- P1 Directory where the CSV import files arestored. This parameter is mandatory.
- P2 Descriptor file required to import the CSV files. This parameter is mandatory. A record descriptor is required to load CSV data into the VSI PERFDAT distributed performance database. A record descriptor is a textual description of all fields (columns) in the CSV file. A record descriptor file can contain 1...n record descriptors. Any CSV file located in the directory specified by P1 that matches the node filter criterion defined by input parameter P5 and that matches one of the record descriptors stored in the file specified by P2 will be loaded by this utility into the VSI PERFDAT distributed performance database. The name of the record descriptor the CSV file refers to has to be encoded in the file name of the CSV file as described below. For more information about record descriptor please refer to the MAP command description in the manual VSI PERFDAT – DQL$ Reference manual.
- P3 Sample interval data stored in the CSV files are collected. This parameter is mandatory.
- P4 CSV format type – MULTI_LINE | SINGLE_LINE. This parameter is mandatory. For more information about record descriptor please refer to the MAP command description in the manual VSI PERFDAT – DQL$ Reference manual.
- P5 Node name
  This parameter is optional. This parameter can be used to filter for CSV files of a specific node (see below).

When the utility is used to load CSV files it checks:

1. If all required parametersare applied. If this is not the case the utility fails.
2. It searches for all CSV files in the directory specified by P1 that matches the filename format:

   Nodename_Metric_YYYY-MM-DD.CSV

   Nodename: Any valid node name the data in the CSV file belongs to. If input parameter P5 has been applied the utility search for these files only that match the node name string applied by P5.

Metric:	Metric name The metric name refers the record descriptor in the descriptor file specified by parameter P2 to be applied to import the CSV file content and it defines the metric name in the target PERFDAT collection database the CSV data is loaded into. If the metric name does not match with any entry in the descriptor file specified by parameter P2 the file is ignored.
YYYY-MM-DD	CSV files contains data of that day
YYYY	Year (e.g. 2006)
MM	Month (e.g. 05 (=May))
DD	Day (e.g. 02) -> leading 0 is required if day is less than 10.

E.g. the CSV file VMSTM1_CPU-2006-03-03.CSV contains data CPU data of node VMSTM1 from date 3-MAY-2006. The appropriate CPU record descriptor has to exist in the descriptor file specified by input parameter P2.

3. If the metric specified in the file name matches an entry in the descriptor file defined by parameter P2 it automatically creates appropriate PERFDAT collection databases and physical storage areas and loads the data of the CSV file into these physical storage areas.
4. After the data load has completed successfully the CSV File is copied to the directory PERFDAT$COMMON:[LOAD.PROCESSED], renamed to *.DONE and deleted from the source directory specified by parameter P1.
5. The processed CSV files have to be deleted manually form directory PERFDAT$COMMON:[LOAD.PROCESSED]:

```
$ DELETE PERFDAT$COMMON:[LOAD.PROCESSED]*.DONE;*
```

The PERFDAT collection databases are referenced by aliases that have the format:

NodeName_CollectionProfile

The collection profile name used by the PERFDAT_LOADCSV utility to create aPERFDAT collection databases is DEFAULT (for more information about PERFDAT database organization please refer to chapter VSI PERFDAT distributed performance databaseof this manual or to the manual VSI PERFDAT – DQL$ Reference manual). E.g. you import CSV data created on node VMSTM1 using this tool the data is loaded into the PERFDAT collection database VMSTM1_DEFAULT.

Top statistics export utility

The utility PERFDAT$TOOLS:DQLGETTOPSTAT.COM can be used to extract the data of the top consuming elements of a statistics from the PERFDAT collection databases. E.g. this utility can be used to search for the top CPU consuming process during a freely definable time interval. Depending on the input parameters applied to the utility the data of the statistics selected of these top elements are either exported to a CSV file or the data of these elements are displayed as line graphs in a PNG file.

Up to seven input parameters can be applied to the utility:

- P1 Mandatory –The command line parameter P1 contains all required data filter parameters:
  - Database alias to attach
  - Metric name
  - Statistics to sort
  - Element filter
  - Start time
  - Stop time
  - Graph mode
    The utility supports two graph modes:
    - un-stacked
    - stacked

    If the user applies the keyword STACKED the selected data will be plotted in stacked mode. If you omit the keyword the data will plotted in un-stacked mode.

  Delimiter symbol to separate the strings = /. You can enter blanks within P1, but in that case apply quotation marks (").
- P2 Optional - Number of elements to be included in top statistics selection output. If you omit the parameter the default of 8 is used.
- P3 Optional - Output type - valid keywords:
  - GRAPH - creates a PNG graphics that includes the data of the selected number of the top elements.
  - CSV - creates a CSV output file that includes the data of the selected number of the top elements.

  If you omit the parameter keyword CSV is assumed.
- P4 Optional - if P3:
  - GRAPH - Name of the HTM and PNG file (see NAME clause description in the CREATE GRAPH command description in the manual VSI PERFDAT – DQL$ Reference manual).
  - CSV - File name of the output file.
- P5 Optional - defines the directory to store the PNG and HTM files in case the output type is GRAPH (parameter P3).
- P6 Optional - regional setting of the output file (see chapter Regional Setting Table in this manual or the DEFINE REGION command description in the manual VSI PERFDAT – DQL$ Reference manual for detailed information about regional settings). This parameter is only valid if P3 = CSV.
- P7 User definable graph header.

If P3 = GRAPH and you omit P4 this command procedure fails.
If P3 = CSV and you omit P4 the output file data of the top elements will be exported to SYS$LOGIN:EXPORT.CSV (see also the EXPORT command description in the manual VSI PERFDAT – DQL$ Reference manual).

If P3 = GRAPH and you omit P5 the output files (PNG, HTM) are created in the graph default directory - PERFDAT$GRAPH (see also the CREATE GRAPH command description in the manual VSI PERFDAT – DQL$ Reference manual).

If P3 = GRAPH and you omit P7 the graph header is automatically created.
If P3 = CSV the command line parameter is ignored.

If parameter P1 is missing the utility fails.

Examples

Example1:

This example demonstrates how to use the utility to export the top four QIO rate DSA devices of node VMSTM1 into a CSV file. The time range of interest is 25-APR-2006 08:00 to 25-APR-2006 18:00. The QIO rate statistics (iQios) is part of the DEVICE metric. The DEVICE metric of node VMSTM1 is stored in the PERFDAT collection database VMSTM1_2MIN. The name of the CSV output file is SYS$LOGIN:DSA_TOP_QIO.CSV. Since P8 is not defined the current region setting is used – in this case DEFAULT.

```
$ @PERFDAT$TOOLS:DQLGETTOPSTAT -
_$ "VMSTM1_2MIN/DEVICE/iQios/DSA*/25-APR-2006 08:00/25-APR-2006 18:00" -
_$ 4 -
_$ CSV -
_$ SYS$LOGIN:DSA_TOP_QIO.CSV
.
.
DQL-I-CFGSUCCESS, default region setting changed to /DEFAULT/
DQL-I-EXPORT, start export data to /SYS$LOGIN:DSA_TOP_QIO.CSV/.
.
.
```

Example2:

The same data are requested as in example 1 but in this case line graphs in a PNG file shall be created in un-stacked mode. The name of the HTM file that refers the PNG file shall be DSA_TOP_QIO.HTM. Since no target directory is defined the PNG file created in the directory PERFDAT$GRAPH. Since parameter P7 is missing the graph header is automatically created.

```
$ @PERFDAT$TOOLS:DQLGETTOPSTAT -
_$ "VMSTM1_2MIN/DEVICE/iQios/DSA*/25-APR-2006 08:00/25-APR-2006 18:00" -
_$ 4 -
_$ GRAPH -
_$ DSA_TOP_QIO
.
.
```

```
.
DQL-GRAPH, Graphs created
PNG file(s):                                    DSA_TOP_QIO.PNG
To view the graphs with your browser access:    DSA_TOP_QIO.HTM
Files are moved to directory:                   PERFDAT$GRAPH:
.
.
.
```

In order to plot the selected data in stacked mode execute:

```
$ @PERFDAT$TOOLS:DQLGETTOPSTAT -
_$ "VMSTM1_2MIN/DEVICE/iQios/DSA*/25-APR-2006 08:00/25-APR-2006 18:00/STACKED" -
_$ 4 -
_$ GRAPH -
_$ DSA_TOP_QIO
.
.
.
DQL-GRAPH, Graphs created
PNG file(s):                                    DSA_TOP_QIO.PNG
To view the graphs with your browser access:    DSA_TOP_QIO.HTM
Files are moved to directory:                   PERFDAT$GRAPH:
```

Brocade switch access test utility

The utility PERFDAT$TOOLS:BROCADE_TEST.COM can be applied to test whether or not a Brocade switch provides valid responses for all OID requests required to run a SNMP performance data collections.

Run this utility before you start any SNMP data collection for your target Brocade switch.

This test utility requires two input parameter:

- P1 IP address or node name of the target system
- P2 Community name
  Enter the SNMPv1/SNMPv2 community name. It is strongly recommended to enter the community name with quotation marks since the SNMP community string check is case sensitive. If you omit quotation marks the community name entered is converted to upper case.
  If P2 is not applied, "FibreChannel" will be used as the default community name for the test.

Example:

To start the Brocade switch access test for the FC switch FCSW201 with community name “public”:

@PERFDAT$TOOLS:BROCADE_TEST FCSW201 “public”

Solaris and Linux configuration test utility

The utility PERFDAT$TOOLS:NET-SNMP_TEST.COM can be applied to test whether or not the NET-SNMP daemon installed on the target system provides valid responses for all OID requests required to run a SNMP performance data collections for Solaris or Linux.

Run this utility before you start any SNMP data collection for your target Solaris or Linux system.

This test utility requires three input parameter:

- P1 target system type
  Valid keywords for P1 are SOLARIS or LINUX. If you want to test the NET-SNMP installation on a Solaris system enter SOLARIS, if you want to test the NET-SNMP installation on a Linux system enter LINUX.
- P2 IP address or node name of the target system
- P3 Community name
  Enter the SNMPv1/SNMPv2 community name you have defined in the *snmpd.conf* file on the target system. It is strongly recommended to enter the community name with quotation marks since the SNMP community string check is case sensitive. If you omit quotation marks the community name entered is converted to upper case.

Example:

To start the NET-SNMP daemon response test for the Linux system PLUTO with community name “perfdat”:

@PERFDAT$TOOLS:NET-SNMP_TEST LINUX PLUTO “perfdat”

APPENDIX A

Statistics available for OpenVMS

ACCOUNT metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
AccountName	Account Name	N/A	
Time	Time	[s]	
iCpuLoad	CPU Load total	[%]	
iKernel	CPU Mode kernel	[%]	
iExec	CPU Mode exec	[%]	
iSuper	CPU Mode super	[%]	
iUser	CPU Mode user	[%]	
iMem	MEM Memory allocated by image	[MB]	
iGlbMem	MEM Gbl Memory allocated by image	[MB]	

iPriMem	MEM Private Memory allocated by image	[MB]
iVAMem	MEM Vitual memory assigned	[MB]
iPfl	PFL total	[1/s]
iPflIO	PFL IO rate	[IO/s]
iPflFOR	PFL on read faults	[1/s]
iPflFOW	PFL on write faults	[1/s]
iPflFOE	PFL on executive faults	[1/s]
iPgflCom	PGFL Pagefile space committed	[MB]
iDIO	IO Direct IO rate	[IO/s]
iBIO	IO Buffered IO rate	[IO/s]
iConCurr	PRC Concurrent processes	[#]
iConCurrUsr	USR Concurrent users	[#]
iCputhres	CPU load threshold	[%]
iMemthres	Memory usage threshold	[MB]
iIOthres	IO request threshold	[IO/s]
iCPUs	CPU number of active CPUs	[#]
iElementCnt	Element count	[#]

CPU metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
CpuID	Cpu ID	N/A	
Time	Time	[s]	
iCpuLoad	CPU Load total	[%]	
iCpuWrkLoad	CPU Work load (= iCpuload - iMpSync)	[%]	
iIntr	Interrupt mode	[%]	
iMPSync	MPSync	[%]	
iKernel	Kernel Mode	[%]	
iExec	Exec Mode	[%]	
iSuper	Super MOde	[%]	
iUser	User Mode	[%]	
iPrcAffCnt	Number of processes with hard affinity to this CPU	N/A	
iCputhres	CPU load threshold	[%]	
iElementCnt	Element count	[#]	

DEVICE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Device	Device Name	N/A	
Time	Time	[s]	
VolumeName	Volume Name	[N/A]	
iQIOs	Total QIO rate on device	[IO/s]	
iRqs	Total device IO request rate	[IO/s]	
iIOs	Total service IO (passing START_IO routine) rate	[IO/s]	
iIOSp	Total split service IO (passing START_IO routine) rate	[IO/s]	
iIOSizeAvg	Average IO size (read & write)	[kB]	V4.8 +
iAbs	Total Aborted	[IO/s]	
iMbs	Total Throughput	[MB/s]	
iRQtime	IO Request time	[ms]	
iRQTimeMax	MAX I/O Request time during last sample interval (all I/Os)	[ms]	V4.1 +
iRQrespAcc	Accuracy of IO Request time	[+/-%]	
iIOtime	IO Service Time	[ms]	
iIOTimeMax	MAX I/O Service time during last sample interval (all I/Os)	[ms]	V4.1 +
iIOrspAcc	Accuracy of IO Service time	[+/-%]	
iRdQIOs	Read QIO rate on device	[IO/s]	
iRdRqs	Read device IO request rate	[IO/s]	
iRdIOs	Read service IO (passing START_IO routine) rate	[IO/s]	

iRdIOSp	Read split service IO (passing START_IO routine) rate	[IO/s]	
iRdIOSizeAvg	Average read IO size	[kB]	V4.8 +
iRdAbs	Read Aborted	[IO/s]	
iRdMbs	Read Throughput	[MB/s]	
iRdRQtime	Read IO Request time	[ms]	
iRdRQTimeMax	Read MAX I/O Request time during last sample interval	[ms]	V4.1 +
iRdRQrespAcc	Accuracy of Read IO Request time	[+/-%]	
iRdIOtime	Read IO Service Time	[ms]	
iIOTimeMax	Read MAX I/O Service time during last sample interval	[ms]	V4.1 +
iRdIOrspAcc	Accuracy of Read IO Service time	[+/-%]	
iWrQIOs	Write QIO rate on device	[IO/s]	
iWrRqs	Write device IO request rate	[IO/s]	
iWrIOs	Write service IO (passing START_IO routine) rate	[IO/s]	
iWrIOSp	Write split service IO (passing START_IO routine) rate	[IO/s]	
iWrIOSizeAvg	Average write IO size	[kB]	V4.8 +
iWrAbs	Write Aborted	[IO/s]	
iWrMbs	Write Throughput	[MB/s]	
iWrRQtime	Write IO Request time	[ms]	
iRQTimeMax	Write MAX I/O Request time during last sample interval)	[ms]	V4.1 +
iWrRQrespAcc	Accuracy of Write IO Request time	[+/-%]	
iWrIOtime	Write IO Service Time	[ms]	
iWrIOTimeMax	Write MAX I/O Service time during last sample interval	[ms]	V4.1 +
iWrIOrspAcc	Accuracy of Write IO Service time	[+/-%]	
iCtlQIOs	Ctrl QIO rate on device	[IO/s]	
iCtlRqs	Ctrl device IO request rate	[IO/s]	
iCtlAbs	Ctrl Aborted	[IO/s]	
iCtlRQtime	Ctrl IO Request time	[ms]	
iRQTimeMax	Ctrl MAX I/O Request time during last sample interval	[ms]	V4.1 +
iCtlRQrespAcc	Accuracy of Ctrl IO Request time	[+/-%]	
iQlen	Device IO queue length	[#]	
iIOthres	IO request threshold	[IO/s]	
iElementCnt	Element count	[#]	

DEVICE.CAPACITY metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Device	Device Name	N/A	
Time	Time	[s]	
VolumeName	Volume Name	[N/A]	
iSize	Device size	[MB]	
iFree	Free space	[MB]	
iUsed	Device space usage	[%]	
iMntCnt	Mount count	[#]	
iErrCnt	Error count	[#]	
iRefCnt	Reference count	[#]	
iSwiAuto	Automatic path switches during last sample period	[1/s]	
iSwiMan	Manual path switches during last sample period	[1/s]	
iPathVer	Number of path verifications during last sample period	[1/s]	
iMVOk	Number of successfull mount verifications	[1/s]	
iMVFail	Number of failed mount verifications	[1/s]	
iElementCnt	Element count	[#]	

DEVICE.FILE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Device	Device Name	N/A	
sFID	FID string	N/A	
Time	Time	[s]	
VolumeName	Volume Name	[N/A]	
iQIOs	Total QIO rate on file	[IO/s]	
iRqs	Total device IO request on file	[IO/s]	
iIOs	Total service IO rate	[IO/s]	
iIOSp	Total split service IO (passing START_IO routine) rate	[IO/s]	
iIOSizeAvg	Average IO size (read & write)	[kB]	V4.8 +
iAbs	Total Aborted	[IO/s]	
iMbs	Total Throughput	[MB/s]	
iRdQIOs	Read QIO rate on file	[IO/s]	
iRdRqs	Read device IO request on file	[IO/s]	
iRdIOs	Read service IO rate	[IO/s]	
iRdIOSp	Read split service IO (passing START_IO routine) rate	[IO/s]	
iRdIOSizeAvg	Average read IO size	[kB]	V4.8 +
iRdAbs	Read Aborted	[IO/s]	
iRdMbs	Read Throughput	[MB/s]	
iWrQIOs	Write QIO rate on file	[IO/s]	

iWrRqs	Write device IO request on file	[IO/s]	
iWrIOs	Write service IO rate	[IO/s]	
iWrIOSp	Write split service IO (passing START_IO routine) rate	[IO/s]	
iWrIOSizeAvg	Average write IO size	[kB]	V4.8 +
iWrAbs	Write Aborted	[IO/s]	
iWrMbs	Write Throughput	[MB/s]	
iCtlQIOs	Ctrl QIO rate on file	[IO/s]	
iCtlRqs	Ctrl device IO request on file	[IO/s]	
iCtlAbs	Ctrl Aborted	[IO/s]	
FID	FID (64 Bit)	N/A	
iIOthres	IO request threshold	[IO/s]	
iElementCnt	Element Count	[#]	

DEVICE.IOSIZE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Device	Device Name	N/A	
IOSize	IO size range	N/A	
Time	Time	[s]	
VolumeName	Volume Name	[N/A]	
iIOs	Total service IO (passing START_IO routine) rate	[IO/s]	
iIOSp	Total split service IO (passing START_IO routine) rate	[IO/s]	
iIOSizeAvg	Average IO size (read & write)	[kB]	V4.8 +
iAbs	Total Aborted	[IO/s]	
iMbs	Total Throughput	[MB/s]	
iIOtime	IO Service Time	[ms]	
iIOrspAcc	Accuracy of IO Service time	[+/-%]	
iRdIOs	Read service IO (passing START_IO routine) rate	[IO/s]	
iRdIOSp	Read split service IO (passing START_IO routine) rate	[IO/s]	
iRdIOSizeAvg	Average read IO size	[kB]	V4.8 +
iRdAbs	Read Aborted	[IO/s]	
iRdMbs	Read Throughput	[MB/s]	
iRdIOtime	Read IO Service Time	[ms]	
iRdIOrspAcc	Accuracy of Read IO Service time	[+/-%]	
iWrIOs	Write service IO (passing START_IO routine) rate	[IO/s]	

iWrIOSp	Write split service IO (passing START_IO routine) rate	[IO/s]	
iWrIOSizeAvg	Average write IO size	[kB]	V4.8 +
iWrAbs	Write Aborted	[IO/s]	
iWrMbs	Write Throughput	[MB/s]	
iWrIOtime	Write IO Service Time	[ms]	
iWrIOrspAcc	Accuracy of Write IO Service time	[+/-%]	
iElementCnt	ElementCnt	[#]	

DEVICE.IOTIMEHIST metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Device	Device Name	N/A	
ReqType	Request type (TOTAL, READ, WRITE)	N/A	
Time	Time	[s]	
VolumeName	Volume Name	N/A	
iIOs	Number of I/Os within last sample interval	[IO]	
iIOtime	Avg. I/O service time during last sample interval	[ms]	
iIOtimeMax	Max I/O service time during last sample interval	[ms]	
2ms	Number of I/Os completed within 2ms	[IO]	
4ms	Number of I/Os completed between 2 and 4ms	[IO]	
6ms	Number of I/Os completed between 4 and 6ms	[IO]	
10ms	Number of I/Os completed between 6 and 10ms	[IO]	
20ms	Number of I/Os completed between 10 and 20ms	[IO]	
30ms	Number of I/Os completed between 20 and 30ms	[IO]	
40ms	Number of I/Os completed between 30 and 40ms	[IO]	
50ms	Number of I/Os completed between 40 and 50ms	[IO]	
100ms	Number of I/Os completed between 50 and 100ms	[IO]	
200ms	Number of I/Os completed between 100 and 200ms	[IO]	
300ms	Number of I/Os completed between 20 and 300ms	[IO]	
400ms	Number of I/Os completed between 300 and 400ms	[IO]	
500ms	Number of I/Os completed between 400 and 500ms	[IO]	
1000ms	Number of I/Os completed between 500 and 1000ms	[IO]	
iGT1sec	Number of I/Os not completed within 1sec	[IO]	

iSampleTime	Sample interval in seconds	[sec]

DEVICE.PATH metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Device	Device Name	N/A	
Path	Device Path	N/A	
Time	Time	[s]	
iOpCnt	Operations completed on Path	[1/s]	
iErrCnt	Error count on path	[1/s]	
isPrimary	Path is primary path	[1/0]	
isCurrent	Path is current path	[1/0]	
isAvail	Path is available	[1/0]	
isResp	Path is responding	[1/0]	
iElementCnt	Element count	[#]	

DEVICE.PROCESS metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Device	Device Name	N/A	
PrcName	Process Name	N/A	
Time	Time	[s]	
VolumeName	Volume Name	N/A	
iQIOs	Total process QIO rate on device	[IO/s]	
iRqs	Total process device IO request on device	[IO/s]	
iIOs	Total service IO rate	[IO/s]	
iIOSp	Total split service IO (passing START_IO routine) rate	[IO/s]	
iIOSizeAvg	Average IO size (read & write)	[kB]	V4.8 +
iAbs	Total Aborted	[IO/s]	
iMbs	Total Throughput	[MB/s]	
iRdQIOs	Read process QIO rate on device	[IO/s]	
iRdRqs	Read process device IO request on device	[IO/s]	
iRdIOs	Read service IO rate	[IO/s]	
iRdIOSp	Read split service IO (passing START_IO routine) rate	[IO/s]	
iRdIOSizeAvg	Average read IO size	[kB]	V4.8 +
iRdAbs	Read Aborted	[IO/s]	
iRdMbs	Read Throughput	[MB/s]	
iWrQIOs	Write process QIO rate on device	[IO/s]	

iWrRqs	Write process device IO request on device	[IO/s]	
iWrIOs	Write service IO rate	[IO/s]	
iWrIOSp	Write split service IO (passing START_IO routine) rate	[IO/s]	
iWrIOSizeAvg	Average write IO size	[kB]	V4.8 +
iWrAbs	Write Aborted	[IO/s]	
iWrMbs	Write Throughpu	[MB/s]	
iCtlQIOs	Ctrl process QIO rate on device	[IO/s]	
iCtlRqs	Ctrl process device IO request on device	[IO/s]	
iCtlAbs	Ctrl Aborted	[IO/s]	
iIOthres	IO request threshold	[IO/s]	
iElementCnt	Element count	[#]	

DEVICE.PROCESS.FILE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Device	Device Name	N/A	
PrcName	Process NameN/A		
sFID	FID string	N/A	
Time	Time	[s]	
VolumeName	Volume Name	N/A	
iQIOs	Total process QIO rate on file	[IO/s]	
iRqs	Total process device IO request on file	[IO/s]	
iIOs	Total service IO rate	[IO/s]	
iIOSp	Total split service IO (passing START_IO routine) rate	[IO/s]	
iIOSizeAvg	Average IO size (read & write)	[kB]	V4.8 +
iAbs	Total Aborted	[IO/s]	
iMbs	Total Throughput	[MB/s]	
iRdQIOs	Read process QIO rate on device	[IO/s]	
iRdRqs	Read process device IO request on device	[IO/s]	
iRdIOs	Read service IO rate	[IO/s]	
iRdIOSp	Read split service IO (passing START_IO routine) rate	[IO/s]	
iRdIOSizeAvg	Average read IO size	[kB]	V4.8 +
iRdAbs	Read Aborted	[IO/s]	
iRdMbs	Read Throughput	[MB/s]	

iWrQIOs	Write process QIO rate on file	[IO/s]	
iWrRqs	Write process device IO request on file	[IO/s]	
iWrIOs	Write service IO rate	[IO/s]	
iWrIOSp	Write split service IO (passing START_IO routine) rate	[IO/s]	
iWrIOSizeAvg	Average write IO size	[kB]	V4.8 +
iWrAbs	Write Aborted	[IO/s]	
iWrMbs	Write Throughput	[MB/s]	
iCtlQIOs	Ctrl process QIO rate on file	[IO/s]	
iCtlRqs	Ctrl process device IO request on file	[IO/s]	
iCtlAbs	Ctrl Aborted	[IO/s]	
FID	FID (64 Bit)	N/A	
iIOthres	IO request threshold	[IO/s]	
iElementCnt	Element count	[#]	

IMAGE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
ImageDsc	Compressed Image Name	N/A	
Time	Time	[s]	
FullImageName	Full Image Name	N/A	
iCpuLoad	CPU Load total	[%]	
iKernel	CPU Mode kernel	[%]	
iExec	CPU Mode exec	[%]	
iSuper	CPU Mode super	[%]	
iUser	CPU Mode user	[%]	
iMem	MEM Memory allocated by image	[MB]	
iGlbMem	MEM Gbl Memory allocated by image	[MB]	
iPriMem	MEM Private Memory allocated by image	[MB]	
iVAMem	MEM Vitual memory assigned	[MB]	
iPfl	PFL total	[1/s]	
iPflIO	PFL IO rate	[IO/s]	
iPflFOR	PFL on read faults	[1/s]	
iPflFOW	PFL on write faults	[1/s]	
iPflFOE	PFL on executive faults	[1/s]	
iPgflCom	PGFL Pagefile space committed	[MB]	
iDIO	IO Direct IO rate	[IO/s]	

iBIO	IO Buffered IO rate	[IO/s]
iConCurr	PRC Concurrent processes running that image	[#]
iCputhres	CPU load threshold	[%]
iMemthres	Memory usage threshold	[MB]
iIOthres	IO request threshold	[IO/s]
iCPUs	CPU number of active CPUs	[#]
iElementCnt	Element count	[#]

IOPATHES metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Path	Device Path	N/A	
Time	Time	[s]	
iOpCnt	Operations completed on IO Path	[1/s]	
iErrCnt	Error count on IO path	[1/s]	
iDevCnt	Number of Devices serviced by IO path	[#]	
iPrimCnt	Number of Devices serviced primary by IO path	[#]	
iCurrCnt	Number of Devices serviced current by IO path	[#]	
iAvailCnt	Number of Devices available via IO Path	[#]	
iRespCnt	Number of Devices responding via IO Path	[#]	
iElementCnt	Element count	[#]	

LANADAPTER metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Adapter	Adapter	N/A	
Time	Time	[s]	
iOctTot	Throughput total	[kB/s]	
iOctRcv	Receive throughput	[kB/s]	
iOctXmt	Transmit throughput	[kB/s]	
iPduTot	PDUs total rate	[1/s]	
iPduRcv	PDUs receive rate	[1/s]	
iPduXmt	PDUs transmit rate	[1/s]	
iMulOctTot	Multicast throughput total	[kB/s]	
iMulOctRcv	Multicast receive throughput	[kB/s]	
iMulOctXmt	Multicast Transmit throughput	[kB/s]	
iMulPduTot	Multicast PDUs total rate	[1/s]	
iMulPduRcv	Multicast PDUs receive rate	[1/s]	
iMulPduXmt	Multicast PDUs transmit rate	[1/s]	
iUnIndDestPdu	Unrecognized individual dest PDU rate	[1/s]	
iUnMulDestPdu	Unrecognized multicast dest PDU rate	[1/s]	
iIniDefPdu	Initially deferred PDU sent rate	[1/s]	
iDataOvrRun	Data overrun rate	[1/s]	
iUnStatBuf	Unavailable station buffer rate	[1/s]	

iUnUsrBuf	Unavailable user buffer rate	[1/s]
iSngColPdu	Single collision PDUs sent rate	[1/s]
iMulColPdu	Multiple collision PDUs sent rate	[1/s]
iExcCol	Excessive collision rate	[1/s]
iCrcErr	CRC error rate	[1/s]
iCcfErr	Carrier check failure rate	[1/s]
iLateCol	Late collision rate	[1/s]
iAlgErr	Alignment error rate	[1/s]
iFrameLong	Frames too long rate	[1/s]
iCollChkFail	Collision detect check failure rate	[1/s]
iFrameSizeErr	Frame size error rate	[1/s]
iXmtLenErr	Send data length error rate	[1/s]
iRcvLenErr	Receive data length error rate	[1/s]
iStatFail	Station failure rate	[1/s]
iShrCirFail	Short circuit failure rate (MOP ONLY)	[1/s]
iOpCirFail	Open circuit failure rate (MOP ONLY)	[1/s]
iXmtLong	Transmit too long rate (MOP ONLY)	[1/s]
iXmtURun	Transmit underrun rate (FDDI ONLY)	[1/s]
iXmtFail	Transmit failure rate (FDDI ONLY)	[1/s]
iFrameStatErr	Frame status error rate (FDDI ONLY)	[1/s]
iFrameLenErr	Frame length error rate (FDDI ONLY)	[1/s]
iMACFrame	MAC frame count rate (FDDI ONLY)	[1/s]
iMACErr	MAC error count rate (FDDI ONLY)	[1/s]
iMACLost	MAC lost count rate(FDDI ONLY)	[1/s]
iRingInit	Ring initializations init rate (FDDI ONLY)	[1/s]
iRingInitsRcv	Ring initializations receive rate (FDDI ONLY)	[1/s]
iRingBeacInit	Ring beacons init rate (FDDI ONLY)	[1/s]
iDuplAddr	Duplicate address test failure rate (FDDI ONLY)	[1/s]
iDupToken	Duplicate tokens detect rate (FDDI ONLY)	[1/s]
iRingPurgeErr	Ring purge error rate (FDDI ONLY)	[1/s]
iFCIStripErr	FCI strip error rate (FDDI ONLY)	[1/s]
iTrace	Traces init rate (FDDI ONLY)	[1/s]
iTraceRcv	Traces receive rate (FDDI ONLY)	[1/s]
iDirBeacRcv	Directed beacons receive rate (FDDI ONLY)	[1/s]
iELBufErr	Elasticity buffer error rate (FDDI ONLY)	[1/s]
iLCTRej	LCT reject rate (FDDI ONLY)	[1/s]
iLEMRej	LEM reject rate (FDDI ONLY)	[1/s]
iLinkErr	Link error rate (FDDI ONLY)	[1/s]
iConnCompl	Connections completed rate (FDDI ONLY)	[1/s]
iOctThres	Octet threshold	[kB/s]
iElementCnt	Element count	[#]

LANADAPTER.DEVICE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Adapter	Adapter	N/A	
Prot./Device	Network Protocol / Virtual LAN device	N/A	
Time	Time	[s]	
iOctTot	Throughput total	[kB/s]	
iOctRcv	Receive throughput	[kB/s]	
iOctXmt	Transmit throughput	[kB/s]	
iPduTot	PDUs total rate	[1/s]	
iPduRcv	PDUs receive rate	[1/s]	
iPduXmt	PDUs transmit rate	[1/s]	
iMulOctTot	Multicast throughput total	[kB/s]	
iMulOctRcv	Multicast receive throughput	[kB/s]	
iMulOctXmt	Multicast Transmit throughput	[kB/s]	
iMulPduTot	Multicast PDUs total rate	[1/s]	
iMulPduRcv	Multicast PDUs receive rate	[1/s]	
iMulPduXmt	Multicast PDUs transmit rate	[1/s]	
iUnaUsrBuf	Unavailable user buffer rate	[1/s]	
iUsrBufSmall	User buffer too small rate	[1/s]	
iElementCnt	Element count	[#]	

LANPROTOCOL metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Protocol	Network Protocol	N/A	
Time	Time	[s]	
iOctTot	Throughput total	[kB/s]	
iOctRcv	Receive throughput	[kB/s]	
iOctXmt	Transmit throughput	[kB/s]	
iPduTot	PDUs total rate	[1/s]	
iPduRcv	PDUs receive rate	[1/s]	
iPduXmt	PDUs transmit rate	[1/s]	
iMulOctTot	Multicast throughput total	[kB/s]	
iMulOctRcv	Multicast receive throughput	[kB/s]	
iMulOctXmt	Multicast Transmit throughput	[kB/s]	
iMulPduTot	Multicast PDUs total rate	[1/s]	
iMulPduRcv	Multicast PDUs receive rate	[1/s]	
iMulPduXmt	Multicast PDUs transmit rate	[1/s]	
iUnaUsrBuf	Unavailable user buffer rate	[1/s]	
iUsrBufSmall	User buffer too small rate	[1/s]	
iElementCnt	Element Count	[#]	

PROCESS metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
PrcName	ProcessName	N/A	
Time	Time	[s]	
PID	Process ID (stored as quad word)	N/A	
UserName	User Name Reference	N/A	
ImageName	Image Name Reference	N/A	
AccountName	Account Name	N/A	
JobName	Job Name Reference in case of Batch process	N/A	
JobMode	Job mode Reference	N/A	
JobType	Job type Reference	N/A	
iCpuLoad	CPU Load total	[%]	
iKernel	CPU Mode kernel	[%]	
iExec	CPU Mode exec	[%]	
iSuper	CPU Mode super	[%]	
iUser	CPU Mode user	[%]	
iMem	MEM Memory allocated by process	[MB]	
iGlbMem	MEM Gbl Memory allocated by process	[MB]	
iPriMem	MEM Private Memory allocated by process	[MB]	
iVAMem	MEM Vitual memory assigned	[MB]	
iPfl	PFL total	[1/s]	

iPflIO	PFL IO rate	[IO/s]
iPflFOR	PFL on read faults	[1/s]
iPflFOW	PFL on write faults	[1/s]
iPflFOE	PFL on executive faults	[1/s]
iPgflQuota	PGFL Page File Quota	[MB]
iPgflFree	PGFL Free Page File Quota	[MB]
iPgflQCons	PGFL Percentage PGFLQUOTA consumption	[%]
iPgflCom	PGFL Pagefile space committed	[MB]
iDIO	IO Direct IO rate	[IO/s]
iBIO	IO Buffered IO rate	[IO/s]
iCPUTim	QUOTA Elapsed CPU time	[s]
iCPUTimInt	QUOTA Elapsed CPU time during last sample interval	[s]
iCPULim	QUOTA CPU time limit	[s]
iDIOCnt	QUOTA Direct I/O count remaining	[#]
iDIOLim	QUOTA Direct I/O limit	[#]
iBIOCnt	QUOTA Buffered I/O count remaining	[#]
iBIOLim	QUOTA Buffered I/O limit	[#]
iBytCnt	QUOTA Buffered I/O byte count remaining	[kB]
iBytLim	QUOTA Buffered I/O byte count limit	[kB]
iAstCnt	QUOTA AST count remaining	[#]
iAstLim	QUOTA AST limit	[#]
iEnqCnt	QUOTA Enqueue count remaining	[#]
iEnqLim	QUOTA Enqueue limit	[#]
iFilCnt	QUOTA Open File count remaining	[#]
iFilLim	QUOTA Open File count limit	[#]
iSPrcCnt	QUOTA Sub-process count remaining	[#]
iSPrcLim	QUOTA Sub-process count limit	[#]
iTqeCnt	QUOTA Timer queue entries remaining	[#]
iTqeLim	QUOTA Timer queue entry limit	[#]
iImgAct	IMG Image activation rate	[1/s]
iCputhres	CPU load threshold	[%]
iMemthres	Memory usage threshold	[MB]
iIOthres	IO request threshold	[IO/s]
iCPUs	CPU number of active CPUs	[#]
iElementCnt	Element count	[#]

SCSPORT metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Port	Port Name	N/A	
Time	Time	[s]	
iMsgRcv	Msg received	[1/s]	
iMsgXmt	Msg sent	[1/s]	
iDGRcv	Datagrams received	[1/s]	
iDGXmt	Datagrams sent	[1/s]	
iBytes	Total throughput (Send & Transmit)	[kB/s]	
iMsgByteRcv	Msg receive throughput	[kB/s]	
iMsgByteXmt	Msg sent throughput	[kB/s]	
iDGByteRcv	Datagram receive throughput	[kB/s]	
iDGByteXmt	Datagram sent throughput	[kB/s]	
iMsgthres	Msg rate threshold	[1/s]	
iElementCnt	Element count	[#]	

SCSPORT.VC metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Port	Port Name	N/A	
VC	Virtual Channel	N/A	
Time	Time	[s]	
iMsg	Msg total	[1/s]	
iBytes	Throughput total	[kB/s]	
iRcvMsg	Rcv total Msg	[1/s]	
iRcvBytes	Rcv throughput	[kB/s]	
iRcvSeq	Rcv sequence Msg	[1/s]	
iRcvUnSeq	Rcv unsequence Msg	[1/s]	
iRcvDupl	Rcv duplicate Msg	[1/s]	
iXmtMsg	Xmt total Msg	[1/s]	
iXmtBytes	Xmt throughput	[kB/s]	
iXmtSeq	Xmt sequence Msg	[1/s]	
iXmtUnSeq	Xmt unsequence Msg	[1/s]	
iXmtAck	Xmt Msg acknowleged	[1/s]	
iXmtReXmt	Xmt retransmit throughput	[kB/s]	
iXmtReXmtReq	Xmt retransmit requests	[1/s]	
iXmtWinFull	Xmt Window full rate	[1/s]	V4.8 +
iAvgRTT	Average round trip delay	[µs]	V4.8 +

iCreditWait	Send Credit wait rate	[1/s]	V4.8 +
iBDLTWait	Buffer descriptor wait rate	[1/s]	V4.8 +
iErrTot	Total Error rate	[1/s]	
iErrTotAbs	Total Errors absolute	[1/s]	
iRcvTRShort	Rcv Error short message	[1/s]	
iRcvIllAck	Rcv Error illegal ack	[1/s]	
iRcvIllSeq	Rcv Error bad sequence	[1/s]	
iRcvBadCks	Rcv Error bad checksum	[1/s]	
iCacheMiss	Rcv Error cache miss	[1/s]	
iFreeQEmty	Rcv Error free queue emty	[1/s]	
iXmtNoXCh	Xmt Error no transmit channel	[1/s]	
iXmtSeqTmo	Xmt Error sequence timeout	[1/s]	
iNoPath	Xmt Error no path	[1/s]	Not avail. for V7.2-2
iXmtLanTmo	Xmt Error LAN timeout	[1/s]	Not avail. for V7.2-2 & V7.3
iNBufSizDec	Error buffer size decrements	[1/s]	Not avail. for V7.2-2
iNPageDynLow	Error low non paged pool	[1/s]	
iMsgthres	Msg rate threshold	[1/s]	
iErrthres	Error rate threshold	[1/s]	
iElementCnt	Element count	[#]	

SCSPORT.VC.CHANNEL metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Port	Port Name	N/A	
VC	Virtual Channel	N/A	
Channel	Channel	N/A	
Time	Time	[s]	
iMsg	Msg total	[1/s]	
iBytes	Throughput total	[kB/s]	
iRcvMsg	Rcv total Msg	[1/s]	
iRcvBytes	Rcv throughput	[kB/s]	
iXmtMsg	Xmt total Msg	[1/s]	
iXmtBytes	Xmt throughput	[kB/s]	
iAvgRTT	Average round trim delay	[µs]	Not valid on V7.2-2
iErrTot	Total Error rate	[1/s]	
iErrTotAbs	Total Errors absolute	[1/s]	
iECSLosses	Error ECS losses	[1/s]	Not valid on V7.2-2
iReXmtPenalty	Xmt Error Retransmit penalty	[1/s]	
iXmtFailPenalty	Xmt Error Transmit failed	[1/s]	
iRcvErrTotal	Rcv Error total	[1/s]	
iRcvBadAuth	Rcv Error bad auth	[1/s]	
iRcvBadEco	Rcv Error bad ECO	[1/s]	

iRcvBadMC	Rcv Error bad MC	[1/s]	
iRcvShortMsg	Rcv Error short Msg	[1/s]	
iRcvIncompChan	Rcv Error Incompatible channel	[1/s]	
iRcvOldMsg	Rcv Error old Msg	[1/s]	
iHSTmo	Error Handshaking timeout	[1/s]	
iListenTmo	Error Listen timeout	[1/s]	
iNoMscpSrv	Error No MSCP server	[1/s]	
iDiskNotSrv	Error No disk served	[1/s]	
iTopChange	Error topology changed	[1/s]	
iPathRestart	Error Path reset	[1/s]	Not valid on V7.2-2
iMsgthres	Msg rate threshold	[1/s]	
iErrthres	Error rate threshold	[1/s]	
iElementCnt	Element count	[#]	

SYSTEM metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V 8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
Node	Node Name	N/A	
Time	Time	[s]	
iCpuLoad	CPU Load total	[%]	
iCpuWrkLoad	CPU Work load (= iCpuload - iMpSync)	[%]	
iIntr	CPU Mode Interrupt	[%]	
iMPSync	CPU Mode MPSync	[%]	
iKernel	CPU Mode Kernel	[%]	
iExec	CPU Mode Exec	[%]	
iSuper	CPU Mode Super	[%]	
iUser	CPU Mode User	[%]	
iCpuCnt	CPU Number of active CPU's	[#]	
iAlignFltKrnl	EXE Alignment faults kernel kernel	[1/s]	Available for: - VSI PERFDAT V4.5 + - OpenVMS V8.3 I64 +
iAlignFltExec	EXE Alignment faults exec mode	[1/s]	Available for: - VSI PERFDAT V4.5 + OpenVMS V8.3 I64 +
iAlignFltSuper	EXE Alignment faults supervisor mode	[1/s]	Available for: - VSI PERFDAT V4.5 + OpenVMS V8.3 I64 +
iAlignFltUserr	EXE Alignment faults user mode	[1/s]	Available for:

			- VSI PERFDAT V4.5 + OpenVMS V8.3 I64 + Available for: - VSI PERFDAT V4.5 + OpenVMS V8.3 I64 +
iMemUseRate	MEM Percentage of memory used	[%]	
iMemUsed	MEM In use	[MB]	
iMemFree	MEM Free	[MB]	
iMemMod	MEM Modified	[MB]	
iMemPhy	MEM Physical memory total	[MB]	
iMemVmsAlloc	MEM Permanently allocated to OpenVMS	[MB]	
iMemReserved	MEM Reserved memory	[MB]	
iNPagCurr	NPAG Current size	[#]	
iNPagInit	NPAG Initial size	[#]	
iNPagMax	NPAG Maximum size	[1/s]	
iNPagFree	NPAG Free space	[1/s]	
iNPagUsed	NPAG Used space	[#]	
iNPagLVarBlk	NPAG Largest Var Block	[#]	
iNPagSVarBlk	NPAG Smallest Var Block	[1/s]	
iNPagFreeBlks	NPAG Number of free blocks	[1/s]	
iNPagVarBlks	NPAG Number of free Blocks (Fragments) in Variable List	[#]	
iNPagLookSize	NPAG Lookaside Space	[#]	
iNPagExpSuc	NPAG Successful Expansions	[#]	
iNPagExpFail	NPAG Failed Expansions	[#]	
iNPagAlloc	NPAG Total Alloc Requests Rate	[1/s]	
iNPagAllocFail	NPAG Failed Alloc Requests Rate	[1/s]	
iCacheInUse	XFC Mem in use	[MB]	Not avail. for V7.2-2
iCacheFree	XFC Mem free	[MB]	Not avail. for V7.2-2
iCacheAlloc	XFC Mem alloc	[MB]	Not avail. for V7.2-2
iCacheHits	XFC Cache Hits	[%]	Not avail. for V7.2-2
iCacheRdHits	XFC Cache Read Hits	[%]	Not avail. for V7.2-2
iCacheRd	XFC Cache Reads	[IO/s]	Not avail. for V7.2-2
iCacheWr	XFC Cache Writes	[IO/s]	Not avail. for V7.2-2
iCacheRdRatio	XFC Cache Read Ratio	[%]	Not avail. for V7.2-2
iCacheRdByp	XFC Reads Bypassed	[IO/s]	Not avail. for V7.2-2
iCacheWrByp	XFC Writes Bypassed	[IO/s]	Not avail. for V7.2-2
iCacheOpFil	XFC Open Files	[#]	Not avail. for V7.2-2
iPflTot	PFL total	[1/s]	
iPflFreeFlt	PFL Free list fault rate	[1/s]	
iPflMdyFlt	PFL Modified list fault rate	[1/s]	
iPflGblPag	PFL global pages	[1/s]	

iPflDZero	PFL Demand Zero	[1/s]	
iPflWrtInPrg	PFL Wrt in progress fault rate	[1/s]	
iPflSysFlt	PFL System fault rate	[1/s]	
iPflRdFlt	PFL read faults	[1/s]	
iPflWrFlt	PFL modified faults	[1/s]	
iPflExecFlt	PFL on executive faults	[1/s]	
iPflReads	PFL total page reads	[IO/s]	
iPflWrites	PFL modified pages written	[IO/s]	
iPflReadIOs	PFL IOs to read pages	[IO/s]	
iPflWriteIOs	PFL IOs to write modified pages	[IO/s]	
iPgflSize	PGFL Total size of all Page File	[MB]	
iPgflFree	PGFL Total free space	[MB]	
iPgflFreeRate	PGFL Percentage of free space	[%]	
iPgflCom	PGFL Committed paging file usage	[MB]	
iPgflIOs	PGFL IOs total	[IO/s]	
iPgflRdIOs	PGFL Read IOs	[IO/s]	
iPgflWrIOs	PGFL Write IOs	[IO/s]	
iIoDIOs	IO DIOs	[IO/s]	
iIoBIOs	IO BIOs	[IO/s]	
iIOSplit	IO Split Ios	[IO/s]	
iDiskIO	IO Disk IOs	[IO/s]	
iDiskMB	IO Disk throughput	[MB]	
iIJobCnt	JOB interactive job count	[#]	
iBJobCnt	JOB batch job count	[#]	
iNJobCnt	JOB network job count	[#]	
iIJobLim	JOB interactive job limit	[#]	
iBJobLim	JOB batch job limit	[#]	
iNJobLim	JOB network job limit	[#]	
iPrcCnt	PRC current process count	[#]	
iPrcMax	PRC maximum process count	[#]	
iPrcAffCnt	PRC number of processes with hard CPU affinity mask set	[#]	
iPrcCre	PRC Process creation rate	[1/s]	Not avail. for V7.2-2 & V7.3
iStateCUR	PRC number of current processes	[#]	
iStateCOM	PRC number of process in COM state	[#]	
iStateCOMO	PRC number of process in COMO state	[#]	
iStateLEF	PRC number of process in LEF state	[#]	
iStateLEFO	PRC number of process in LEFO state	[#]	
iStateHIB	PRC number of process in HIB state	[#]	
iStateHIBO	PRC number of process in HIBO state	[#]	
iStateCEF	PRC number of processes in CEF state	[#]	

iStateSUSP	PRC number of processes in SUSP state	[#]
iStateSUSPO	PRC number of processes in SUSPO state	[#]
iStateMWAIT	PRC number of processes in MWAIT state	[#]
iStateCOLPG	PRC number of processes in COLPAG (collided page wait) state	[#]
iStatePFW	PRC number of processes in PFW (pagefault wait) state	[#]
iStateFPG	PRC number of processes in FPG (free page wait) state	[#]
iMscpVCFail	MSCP VC Failures	[1/s]
iMscpIOs	MSCP Total IOs	[IO/s]
iMscpSplitIO	MSCP Split IOs	[IO/s]
iMscpIOWait	MSCP buffer wait rate	[1/s]
iXQPCall	XQP FCP Call Rate	[1/s]
iXQPAlloc	XQP Allocation Rate	[1/s]
iXQPCreate	XQP Create Rate	[1/s]
iXQPDskRd	XQP Disk Read Rate	[1/s]
iXQPDskWr	XQP Disk Write Rate	[1/s]
iXQPVolLckWait	XQP Volume sync. Lock Wait Rate	[1/s]
iXQPFilLckWait	XQP Directory & File sync. Lock Wait Rate	[1/s]
iXQPLookup	XQP File Lookup Rate	[1/s]
iXQPOpen	XQP Current Open Files	[#]
iXQPOpenRate	XQP File Open Rate	[1/s]
iXQPErase	XQP Erase QIO Rate	[1/s]
iXQPFhhCHit	XQP File Header Cache Hits	[1/s]
iXQPFhCMiss	XQP File Header Cache Misses	[1/s]
iXQPFhCHRact	XQP actual File Header Cache Hit Rate	[%]
iXQPFhCHRall	XQP overall File Header Cache Hit Rate	[%]
iXQPSbCHit	XQP Storage Bitmap Cache Hits	[1/s]
iXQPSbCMiss	XQP Storage Bitmap Cache Misses	[1/s]
iXQPSbCHRact	XQP actual Storage Bitmap Cache Hit Rate	[%]
iXQPSbCHRall	XQP overall Storage Bitmap Cache Hit Rate	[%]
iXQPDdCHit	XQP Directory Data Cache Hits	[1/s]
iXQPDdCMiss	XQP Directory Data Cache Misses	[1/s]
iXQPDdCHRact	XQP actual Directory Data Cache Hit Rate	[%]
iXQPDdCHRall	XQP overall Directory Data Cache Hit Rate	[%]
iXQPFidCHit	XQP FID Cache Hits	[1/s]
iXQPFidCMiss	XQP FID Cache Misses	[1/s]
iXQPFidCHRact	XQP actual FID Cache HitRate	[%]
iXQPFidCHRall	XQP overall FID Cache HitRate	[%]
iXQPExtCHits	XQP Extent Cache Hits	[1/s]
iXQPExtCMiss	XQP Extent Cache Misses	[1/s]

iXQPExtCHRact	XQP actual Extent Cache HitRate	[%]	
iXQPExtCHRall	XQP overall Extent Cache HitRate	[%]	
iXQPWinTurn	XQP Window Turns	[1/s]	
iLckLoc	LCK percentage of local locking activity	[%]	
iLckOut	LCK percentage of remote locking activity	[%]	
iLckENQnewloc	LCK new ENQ local	[1/s]	
iLckENQnewin	LCK new ENQ in	[1/s]	
iLckENQnewout	LCK new ENQ out	[1/s]	
iLckENQcvloc	LCK ENQ converts local	[1/s]	
iLckENQcvin	LCK ENQ converts in	[1/s]	
iLckENQcvout	LCK ENQ converts out	[1/s]	
iLckDEQloc	LCK DEQ local	[1/s]	
iLckDEQin	LCK DEQ in	[1/s]	
iLckDEQout	LCK DEQ out	[1/s]	
iLckENQwait	LCK ENQ waits	[1/s]	
iLckENQnoQue	LCK ENQ not queued	[1/s]	
iLckDirIn	LCK directory ops in	[1/s]	
iLckDirOut	LCK directory ops out	[1/s]	
iLckDLfind	LCK dead lock find rate	[1/s]	
iLckDLsearch	LCK dead lock search rate	[1/s]	
iDLckMsgIn	LCK distributed Msg in	[1/s]	
iDlckMsgOut	LCK distributed Msg out	[1/s]	
iLckRMunload	LCK remaster unload rate	[1/s]	
iLckRMacq	LCK remaster acquire rate	[1/s]	
iLckRMfinish	LCK remaster finish rate	[1/s]	
iLckRMMsgSent	LCK remaster Msg sent	[1/s]	
iLckRMMsgRcv	LCK remaster Msg rcv	[1/s]	
iLckRMRbldSent	LCK remaster Rebuild sent	[1/s]	
iLckRMRbldRcv	LCK remaster Rebuild rcv	[1/s]	
iLckRMReqAck	LCK remaster Req not ack	[1/s]	
iLckRMquotWait	LCK remaster quota wait	[1/s]	
iLckRMbetter	LCK remastered – higher LCKDIRWAIT	[1/s]	
iLckRMsngl	LCK remastered - single node interest	[1/s]	
iLckRMact	LCK remastered – higher Lck Activity	[1/s]	
iTQEtot	TQE total	[1/s]	
iTQEsysup	TQE sysup	[1/s]	
iTQEtimer	TQE timer	[1/s]	
iTQEschwk	TQE scheduled wakeup	[1/s]	
iLNMtran	LNM translations	[1/s]	Not avail. for V7.2-2
iLNMtranFail	LNM translations fail	[1/s]	Not avail. for V7.2-2
iLNMcreate	LNM logical name creation	[1/s]	Not avail. for V7.2-2

iLNMdelete	LNM logical name deletion	[1/s]	Not avail. for V7.2-2
iElementCnt	ElementCnt	[#]	

USER metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.2-2 Alpha	V3.3
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
UserName	User Name	N/A	
Time	Time	N/A	
AccountName	Account Name Reference	N/A	
iCpuLoad	CPU Load total	[%]	
iKernel	CPU Mode kernel	[%]	
iExec	CPU Mode exec	[%]	
iSuper	CPU Mode super	[%]	
iUser	CPU Mode user	[%]	
iMem	MEM Memory allocated by user	[MB]	
iGlbMem	MEM Gbl Memory allocated by user	[MB]	
iPriMem	MEM Private Memory allocated by user	[MB]	
iVAMem	MEM Vitual memory assigned	[MB]	
iPfl	PFL total	[1/s]	
iPflIO	PFL IO rate	[IO/s]	
iPflFOR	PFL on read faults	[1/s]	
iPflFOW	PFL on write faults	[1/s]	
iPflFOE	PFL on executive faults	[1/s]	
iPgflCom	PGFL Pagefile space committed	[MB]	
iDIO	IO Direct IO rate	[IO/s]	

iBIO	IO Buffered IO rate	[IO/s]
iImgAct	IMG Image activation rate	[1/s]
iConCurr	PRC Concurrent processes	[#]
iCputhres	CPU load threshold	[%]
iMemthres	Memory usage threshold	[MB]
iIOthres	IO request threshold	[IO/s]
iCPUs	CPU number of active CPUs	[#]
iElementCnt	Element count	[#]

XFCVOLUME metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
VolumeName	VolumeName	N/A	
Time	Time	[s]	
iQIOs	QIOs total	[IO/s]	
iReads	Read IOs	[IO/s]	
iWrites	Write IOs	[IO/s]	
iReadAhead	Read ahead rate	[IO/s]	
iReadAround	Read around rate	[IO/s]	
iWriteAround	Write around rate	[IO/s]	
iWriteBehind	Write behind rate	[IO/s]	
iWriteLost	Rate of writes lost due to disk error	[IO/s]	
iHits	Hit rate in last sample interval	[%]	
iMiss	Cache miss rate in last sample interval	[%]	V3.3 ECO 1 +
iHitsAvg	Average Hit rate	[%]	
iReadMB	Read throughput	[MB/s]	
iWriteMB	Write throughput	[MB/s]	
iThruAcc	Throughput accuracy	[%]	
iRspHit	Cache Hit response time	[ms]	
iRspMiss	Cache miss response time	[ms]	
iRspTotal	Overall response time	[ms]	
iRspAcc	Response time accuracy	[%]	

iCacheMB	Cache Memory used by this volume	[MB]
iOpenFiles	Actual open Files on Volume	[#]
iClosedFiles	Actual closed Files on Volume	[#]
iOpenFilRate	File open rate	[1/s]
iCloseFilRate	File close rate	[1/s]
iIOthres	IO request threshold	[IO/s]
iElementCnt	Element count	[#]

XFCVOLUME.IOSIZE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
VolumeName	VolumeName	N/A	
IOSize	IO size Range	N/A	
Time	Time	[s]	
iQIOs	QIOs total	[IO/s]	
iRdIO	Read requests	[IO/s]	
iWrIO	Write requests	[IO/s]	
iHits	Hit rate in last sample interval	[%]	
iMiss	Cache miss rate in last sample interval	[%]	V3.3 ECO 1 +
iHitsAvg	Average hit rate	[%]	
iRdMB	Read throughput	[MB]	
iWrMB	Write throughput	[MB]	
iElementCnt	Element count	[#]	

XFCVOLUME.FILE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
VolumeName	VolumeName	N/A	
sFID	FID string	N/A	
Time	Time	[s]	
iQIOs	QIOs total	[IO/s]	
iReads	Read IOs	[IO/s]	
iWrites	Write IOs	[IO/s]	
iHits	Hit Rate in last sample interval	[%]	
iMiss	Cache miss rate in last sample interval	[%]	V3.3 ECO 1 +
iHitsAvg	Average Hit rate	[%]	
iReadMB	Read throughput	[MB/s]	
iWriteMB	Write throughput	[MB/s]	
iThruAcc	Throughput accuracy	[%]	
iReadAhead	Read ahead rate	[IO/s]	
iReadAround	Read around rate	[IO/s]	
iReadThrough	Read Through rate	[IO/s]	
iWriteBehind	Write behind rate	[IO/s]	
iWriteAround	Write around rate	[IO/s]	
iWriteThrough	Write Through rate	[IO/s]	
iCacheMB	Cache memory used by this file	[MB]	
FID	FID (64 Bit)	N/A	

iIOthres	IO request threshold	[IO/s]
iElementCnt	Element count	[#]

XFCVOLUME.FILE.IOSIZE metric

Available for:

OpenVMS Version	VSI PERFDAT Version
V7.3 Alpha	V3.3
V7.3-1 Alpha	V3.3
V7.3-2 Alpha	V4.8
V8.2 Alpha	V4.8
V8.3 Alpha	V4.8
V8.4 Alpha	V4.8
V8.2 I64	V4.8
V8.2-1 I64	V4.8
V8.3 I64	V4.8
V8.3-1H1 I64	V4.8
V8.4 I64	V4.8
V8.4-1H1 I64	V4.8
V8.4-2 I64	V4.8
V8.4-2L1 I64	V4.8

Statistics	Description	Unit	Comments
VolumeName	VolumeName	N/A	
sFID	FID string	N/A	
IOSize	IO size Range	N/A	
Time	Time	[s]	
iQIOs	QIOs total	[IO/s]	
iRdIO	Read requests	[IO/s]	
iWrIO	Write requests	[IO/s]	
iHits	Hit rate in last sample interval	[%]	
iMiss	Cache miss rate in last sample interval	[%]	V3.3 ECO 1 +
iHitsAvg	Average hit rate	[%]	
iRdMB	Read throughput	[MB]	
iWrMB	Write throughput	[MB]	
FID	FID (64 Bit)	N/A	
iElementCnt	ElementCnt	[#]	

Statistics available for HP StorageWorks Virtual Arrays

ARRAY metric

Statistics	Description	Unit	Comments
Name	System Name	N/A	
Time	Time	[s]	
iCpuLoad	CPU load	[%]	
iReqs	Total Requests (Read & Write)	[1/s]	
iRdReqs	Read Requests	[1/s]	
iWrReqs	Write Requests	[1/s]	
iKb	Total Throughput (Read & Write)	[kB/s]	
iRdKb	Read Throughput	[kB/s]	
iWrKb	Write Throughput	[kB/s]	
iSize	Average I/O Size (Read & Write)	[kB]	
iRdSize	Average Read I/O Size	kB]	
iWrSize	Average Write I/O Size	[kB]	
iLatency	Average I/O Latency (Read & Write)	[us]	
iRdLatency	Average Read I/O Latency	[us]	
iWrLatency	Average Write I/O Latency	[us]	
iQue	Average Queue Length[#]		

CTRL metric

Statistics	Description	Unit	Comments
Ctrl	Controller Name	N/A	
Time	Time	[s]	
iCpuLoad	CPU load	%]	
iReqs	Total Requests (Read & Write)	[1/s]	
iRdReqs	Read Requests	[1/s]	
iWrReqs	Write Requests	[1/s];	
iKb	Total Throughput (Read & Write)	[kB/s]	
iRdKb	Read Throughput	[kB/s]	
iWrKb	Write Throughput	[kB/s]	
iSize	Average I/O Size (Read & Write)	[kB]	
iRdSize	Average Read I/O Size	[kB]	
iWrSize	Average Write I/O Size	[kB]	
iLatency	Average I/O Latency (Read & Write)	[us]	
iRdLatency	Average Read I/O Latency	[us]	
iWrLatency	Average Write I/O Latency	[us]	
iQue	Average Queue Length[#]		

CTRL.PORT metric

Statistics	Description	Unit	Comments
Ctrl	Controller Name	N/A	
Port	Port Name	N/A	
Time	Time	[s]	
iReqs	Total Requests (Read & Write)	[1/s]	
iRdReqs	Read Requests	[1/s]	
iWrReqs	Write Requests	[1/s];	
iKb	Total Throughput (Read & Write)	[kB/s]	
iRdKb	Read Throughput	[kB/s]	
iWrKb	Write Throughput	[kB/s]	
iSize	Average I/O Size (Read & Write)	[kB]	
iRdSize	Average Read I/O Size	[kB]	
iWrSize	Average Write I/O Size	[kB]	
iLatency	Average I/O Latency (Read & Write)	[us]	
iRdLatency	Average Read I/O Latency	[us]	
iWrLatency	Average Write I/O Latency	[us]	
iQue	Average Queue Length[#]		

CTRL.HOSTCONN metric

Statistics	Description	Unit	Comments
Ctrl	Controller Name	N/A	
HostName	Host Name	N/A	
HostWWN	Host Connection WWN	N/A	
Time	Time	[s]	
sID	Internal HOST Connection ID	N/A	
iQue	Average Queue Length	[#]	
iBusy	Busy States	[#]	

DISKGROUP metric

Statistics	Description	Unit	Comments
Name	Disk Group Name	N/A	
Time	Time	[s]	
iCapacity	Disk Group Capacity/Size	[GB]	
iOccupancy	Disk Group Capacity occupied	[GB]	
iOccUsed	Disk Group Capacity used	[%]	
iOccHigh	Disk Group Occupancy High-water Mark	[%]	
iDisks	Number of disks in the Disk Group	[#]	
iSpareCurr	Spare disks available	[#]	
iSpareGoal	Spare disks configured	[#]	
iVDReqs	VDisk summary Total Requests (Read & Write)	[1/s]	
iVDRdReqs	VDisk summary Read Requests	[1/s]	
iVDRdHitReqs	VDisk summary Read Hit Requests	[1/s]	
iVDRdMissReqs	VDisk summary Read Miss Requests	[1/s]	
iVDWrReqs	VDisk summary Write Request	[1/s]	
iVDRdRatio	VDisk summary Read Ratio (iVDRdReqs / iVDReqs)	[%]	
iVDRdHitRatio	VDisk summary Read Hit Ratio (iVDRdHitReqs / iVDRdReqs)	[%]	
iVDMB	VDisk summary Total Throughput (Read & Write)	[MB/s]	
iVDRdMB	VDisk summary Read Throughput	[MB/s]	
iVDRdHitMB	VDisk summary Read Hit Throughput	[MB/s]	
iVDRdMissMB	VDisk summary Read Miss Throughput	[MB/s]	
iVDWrMB	VDisk summary Write Throughput	[MB/s]	
iVDFlushMB	VDisk summary Data Flushed	[MB/s]	
iVDPreFetchMB	VDisk summary Data Prefetched	[MB/s]	
iVDMirrorMB	VDisk summary Data Mirrored	[MB/s]	
iPDReqs	PDisk summary Total Requests (Read & Write)	[1/s]	
iPDRdReqs	PDisk summary Read Request	[1/s]	
iPDWrReqs	PDisk summary Write Request	[1/s]	
iPDMB	PDisk summary Total Throughput (Read & Write)	[MB/s]	
iPDRdMB	PDisk summary Read Throughput	[MB/s]	
iPDWrMB	PDisk summary Write Throughput	[MB/s]	

DISKGROUP.PDISK metric

This metric is not available if performance data are collected from HSV 240 or HSV360 controllers (EVA 6300/6350/6500/6550).

Statistics	Description	Unit	Comments
Group	Disk Group Name	N/A	
sName	Diks Name	N/A	
Time	Time	[s]	
iReqs	Total Requests (Read & Write)	[1/s]	
iRdReqs	Read Request	[1/s]	
iWrReqs	Write Request	[1/s];	
iKb	Total Throughput (Read & Write)	[kB/s]	
iRdKb	Read Throughput	[kB/s]	
iWrKb	Write Throughput	[kB/s]	
iSize	Average I/O Size (Read & Write)	[kB]	
iRdSize	Average Read I/O Size	[kB]	
iWrSize	Average Write I/O Size	[kB]	
iLatency	Average Drive Latency	[us]	
iQue	Average Drive Queue Length	[#]	
iShelf	ShelfN/A		
iBay	BayN/A		

DISKGROUP.VDISK metric

Statistics	Description	Unit	Comments
Group	Disk Group Name	N/A	
Device	Device Name	N/A	
Time	Time	[s]	
sUUID	Device Name	N/A	
iReqs	Total Requests (Read & Write)	[1/s]	
iRdReqs	Read Requests	[1/s]	
iRdHitReqs	Read Hit Requests	[1/s]	
iRdMissReqs	Read Miss Requests	[1/s]	
iWrReqs	Write Request	[1/s];	
iRdRatio	Read Ratio (iRdReqs / iReqs)	[%]	
iRdHitRatio	Read Hit Ratio (iRdHitReqs / iRdReqs)	[%]	
iKb	Total Throughput (Read & Write)	[kB/s]	
iRdKb	Read Throughput	[kB/s]	
iRdHitKb	Read Hit Throughput	[kB/s]	
iRdMissKb	Read Miss Throughput	[kB/s]	
iWrKb	Write Throughput	[kB/s]	
iFlushKb	Data Flushed	[kB/s]	
iPreFetchKb	Data Prefetched	[kB/s]	
iMirrorKb	Data Mirrored	[kB/s]	
iSize	Average I/O Size (Read & Write)	[kB]	
iRdSize	Average Read I/O Size	[kB]	
iRdHitSize	Average Read Hit I/O Size	[kB]	
iRdMissSize	Average Read Miss I/O Size	[kB]	
iWrSize	Average Write I/O Size	[kB]	
iLatency	Average I/O Latency (Read & Write)	[us]	
iRdLatency	Average Read I/O Latency	[us]	
iRdHitLat	Average Read Hit I/O Latency	[us]	
iRdMissLat	Average Read Miss I/O Latency	[us]	
iWrLatency	Average Write I/O Latency	[us]	

DRM.TUNNEL metric

Statistics	Description	Unit	Comments
Ctrl	Controller Name	N/A	
Tunnel	Tunnel	N/A	
Time	Time	[s]	
iRtDly	Roundtrip delay	[ms]	
iCpyIn	Copy IN	[kB/s]	
iCpyOut	Copy OUT	[kB/s]	
iWrtIn	Write IN	[kB/s]	
iWrtOut	Write OUT	[kB/s]	
iCpyRetries	Copy Retries	[1/s]	
iWrtRetries	Write Retries	[1/s]	
iCpyMinRes	Min Copy Resources	[#]	
iWrtMinRes	Min Write Resources	[#]	
iCmdMinRes	MIn Command Resources	[#]	

Statistics available for Tru64

TRU64_CPU metric

Statistics	Description	Unit	Comments
CpuId	Cpu Index	N/A	
Time	Time	[s]	
CpuIdle	CPU Cpu utilization idle state	[%]	
CpuLoad	CPU Cpu utilization	[%]	
CpuUser	CPU Cpu utilization user state	[%]	
CpuSystem	CPU Cpu utilization system state	[%]	
CpuNice	CPU Cpu utilization nice state	[%]	
CpuWait	CPU Cpu utilization wait state	[%]	

TRU64_DEAMON metric

Statistics	Description	Unit	Comments
StartCmd	Deamon / start cmd	N/A	
Time	Time	[s]	
User	User name	N/A	
CpuLoad	CPU average Cpu load during last sample interval	[%]	
CpuAct	CPU load current	[%]	
MemVirt	MEM virtual mem size of the process	[MB]	
MemResSet	MEM resident set size of the process	[MB]	
MajFlt	MEM major page fault rate	[1/s]	
InBlk	IO block input operation rate	[1/s]	
OutBlk	IO block output operation rate	[1/s]	
ThrdCnt	Thread count	[#]	

TRU64_DISK metric

Statistics	Description	Unit	Comments
Name	Block device name	N/A	
Time	Time	[s]	
DevRate	DEV transfer rate	[1/s]	
DevKB	DEV throughput	[kB/s]	
DevWaitIO	DEV avg. I/O wait time	[ms]	
DevWaitTimAbs	DEV accum. waittime during sample interval	[us]	
DevWaitTim	DEV accum. waittime during sample interval	[ms]	
DevSrvIO	DEV avg. I/O service time	[ms]	
DevSrvTimAbs	DEV accum. service time during sample interval	[us]	
DevIOPendAbs	DEV number of I/Os in pending queue during sample	[#]	
DevSrvTim	DEV accum. service time during sample interval	[ms]	
DevIOPend	DEV average number of I/Os in pending queue	[1/s]	
DevQue	DEV queue length	[#]	

TRU64_FILESYS metric

Statistics	Description	Unit	Comments
Name	File System name	N/A	
Time	Time	[s]	
Size	Total Size	[GB]	
Free	Free Size	[GB]	
Used	Used Size	[GB]	
Avail	Available Size	[GB]	

TRU64_IP metric

Statistics	Description	Unit	Comments
Name	Protocol Name	N/A	
Time	Time	[s]	
iDGTot	Total Datagram rate (in & out)	[1/s]	
iDGRcv	Datagram receive rate	[1/s]	
iDGXmt	Datagram transmit rate	[1/s]	
iRcvDeliver	IN Rate of successful datagram delivery to IP-stack	[1/s]	
iRcvForward	IN Forwarding datagram rate	[1/s]	
iRcvHdrErr	IN IP-Header error rate	[1/s]	
iRcvAddrErr	IN Address error rate	[1/s]	
iRcvUnkProt	IN Unknown protocol rate	[1/s]	
iRcvDiscard	IN Datagram discard rate	[1/s]	
iXmtNoRout	OUT Datagram discard rate	[1/s]	
iXmtDiscard	OUT Datagram discard due to no route	[1/s]	

TRU64_NIC metric

Statistics	Description	Unit	Comments
Interface	Interface Name	N/A	
Time	Time	[s]	
OctTot	TOT total number of octets (rcv & tmx)	[kB/s]	
UCasts	TOT total number of subnet-unicast packets	[1/s]	
NUCast	TOT total number of non-unicast packes (e.g. broadcasts)	[1/s]	
Discards	TOT total number of packets discarded	[1/s]	
ErrTot	TOT total errors on interface	[1/s]	
InOct	RCV octets received	[kB/s]	
InUCast	RCV number of subnet-unicast packets	[1/s]	
InNUCast	RCV number of non-unicast packets (e.g. broadcasts)	[1/s]	
InDiscards	RCV inbound msg discarded without error	[1/s]	
InUnkProt	RCV inbound msg discarded because of unknown protocol	[1/s]	
InErr	RCV inbound msg with error	[1/s]	
OutOct	TMX octets transmitted	[kB/s]	
OutUCast	TMX number of subnet-unicast packets requests	[1/s]	
OutNUCast	TMX number of non-unicast packet requests (e.g. broadcasts)	[1/s]	
OutDiscards	TMX outbound msg discarded without error	[1/s]	
OutErr	TMX outbound msg with error	[1/s]	
OutQue	TMX output packet queue	[#]	

TRU64_PROCESS metric

Statistics	Description	Unit	Comments
StartCmd	Command that started the process	N/A	
Pid	Process ID	N/A	
Time	Time	[s]	
User	User name	N/A	
CpuLoad	CPU average Cpu load during last sample interval	[%]	
CpuAct	CPU load current	[%]	
MemVirt	MEM virtual mem size of the process	[MB]	
MemResSet	MEM resident set size of the process	[MB]	
MajFlt	MEM major page fault rate	[1/s]	
InBlk	IO block input operation rate	[1/s]	
OutBlk	IO block output operation rate	[1/s]	
ThrdCnt	Thread count	[#]	
BasePrio	Base priority of the process	[#]	
Nice	Process scheduling increment	[#]	

TRU64_SYSTEM metric

Statistics	Description	Unit	Comments
Node	OS name	N/A	
Time	Time	[s]	
CpuIdle	CPU total idle state utilization	[%]	
CpuLoad	CPU total load	[%]	
CpuUser	CPU total user state utilization	[%]	
CpuSystem	CPU total system state utilization	[%]	
CpuNice	CPU total nice state utilization	[%]	
CpuWait	CPU total wait state utilization	[%]	
DevIntr	CPU device interrupt rate	[1/s]	
SysCalls	CPU system call rate	[1/s]	
CtxSwitch	CPU context switch rate	[1/s]	
CpuCnt	CPU number of CPUs online		
PhyMemSize	MEM physical memory available	[MB]	
PhyMemUse	MEM physical memory currently used	[MB]	
VirMemFree	MEM free memory	[MB]	
VirMemAct	MEM active page-able memory	[MB]	
VirMemInAct	MEM inactive page-able memory	[MB]	
VirMemWire	MEM non page-able (wired) kernel memory	[MB]	
SwapTot	MEM total swap space	[MB]	
SwapFree	MEM free swap space	[MB]	
SwapUsed	MEM used swap space	[MB]	
PflTot	PFL page fault rate	[1/s]	
PflPagIn	PFL pages paged in rate	[MB/s]	
PflPagOut	PFL pages paged out rate	[MB/s]	
PrcTot	PRC total number of process	[#]	
PrcSwapIn	PRC number of process swapped in	[#]	
PrcSwapOut	PRC number of process swapped out	[#]	

TRU64_USER metric

Statistics	Description	Unit	Comments
User	User	N/A	
Time	Time	[s]	
CpuLoad	CPU average Cpu load during last sample interval	[%]	
CpuAct	CPU load current	[%]	
MemVirt	MEM virtual mem size of the process	[MB]	
MemResSet	MEM resident set size of the process	[MB]	
MajFlt	MEM major page fault rate	[1/s]	
InBlk	IO block input operation rate	[1/s]	
OutBlk	IO block output operation rate	[1/s]	
ThrdCnt	Thread count	[#]	

Statistics available for Brocade switches

PORT metric

Statistics	Description	Unit	Comments
Index	Identifies the switch port index	N/A	
Name	Port Name	N/A	V4.1 +
Time	Time	[s]	
PortSts	Identifies the physical state of the port	[#]	
OpStatus	Identifies the operational status of the port	[#]	
TxType	This object indicates the media transmitter type of the port	[#]	
TotWords	Throughput on port (Read & Write)	[kB/s]	
TxWords	Transmit throughput on port	[kB/s]	
RxWords	Receive throughput on port	[kB/s]	
TotFrm	Fibre Channel frame rate	[1/s]	
TxFrm	Fibre Channel frame transmit rate	[1/s]	
RxFrm	Fibre Channel frame receive rate	[1/s]	
RxC2Frm	Class 2 frame receive rate	[1/s]	
RxC3Frm	Class 3 frame receive rate	[1/s]	
RxLCs	Link Control frame receive rate	[1/s]	
RxMcasts	Multicast frame receive rate	[1/s]	
TooManyRdys	Rate of RDYs exceeds the frames received	[1/s]	
NoTxCredits	Rate of transmit credit has reached zero	[1/s]	
RxEncInFrs	Rate of encoding error or disparity error inside frames received	[1/s]	
RxCrcs	Rate of CRC errors detected for frames received	[1/s]	
RxTruncs	Rate of truncated frames that the port has received	[1/s]	
RxTooLongs	Rate of received frames that are too long	[1/s]	
RxBadEofs	Rate of received frames that have bad EOF delimiter	[1/s]	
RxEncOutFrs	Rate of encoding error or disparity error outside frames received	[1/s]	
RxBadOs	Rate of invalid Ordered Sets received	[1/s]	
C3Discard	Rate of Class 3 frames that the port has discarded	[1/s]	
McastTimedOut	Rate of Multicast frames that has been timed out	[1/s]	
TxMcasts	Rate of Multicast frames that has been transmitted	[1/s]	
LipIns	Rate of Loop Initializations that has been initiated by loop devices attached	[1/s]	
LipOuts	Rate of Loop Initializations that has been initiated by the port	[1/s]	

SYSTEM metric

Statistics	Description	Unit	Comments
Name	Identifies the switch	N/A	
Time	Time	[s]	
TotWords	Switch Throughput (Read & Write)	[k/s]	
TxWords	Switch transmit throughput	[kB/s]	
RxWords	Switch Receive throughput	[kB/s]	
TotFrm	Fibre Channel frame rate	[1/s]	
TxFrm	Fibre Channel frame transmit rate	[1/s]	
RxFrm	Fibre Channel frame receive rate	[1/s]	
RxC2Frm	Class 2 frame receive rate	[1/s]	
RxC3Frm	Class 3 frame receive rate	[1/s]	
RxLCs	Link Control frame receive rate	[1/s]	
RxMcasts	Multicast frame receive rate	[1/s]	
TooManyRdys	Rate of RDYs exceeds the frames received	[1/s]	
NoTxCredits	Rate of transmit credit has reached zero	[1/s]	
RxEncInFrs	Rate of encoding error or disparity error inside frames received	[1/s]	
RxCrcs	Rate of CRC errors detected for frames received	[1/s]	
RxTruncs	Rate of truncated frames that the port has received	[1/s]	
RxTooLongs	Rate of received frames that are too long	[1/s]	
RxBadEofs	Rate of received frames that have bad EOF delimiter	[1/s]	
RxEncOutFrs	Rate of encoding error or disparity error outside frames received	[1/s]	
RxBadOs	Rate of invalid Ordered Sets received	[1/s]	
C3Discard	Rate of Class 3 frames that the port has discarded	[1/s]	
McastTimedOut	Rate of Multicast frames that has been timed out	[1/s]	
TxMcasts	Rate of Multicast frames that has been transmitted	[1/s]	
LipIns	Rate of Loop Initializations that has been initiated by loop devices attached	[1/s]	
LipOuts	Rate of Loop Initializations that has been initiated by the port	[1/s]	

SYSTEM.FAN metric

Statistics	Description	Unit	Comments
Name	Sensor Name	N/A	
Time	Time	[s]	
Type	Sensor Type (informational only – not displayed in the GUI)	N/A	
Status	Status (1=Unknown, 2=Faulty, 3=Below-min, 4=nominal, 5=above-max)	N/A	
FanSpeed	Fan Speed	[RPM]	

SYSTEM.TEMPERATURE metric

Statistics	Description	Unit	Comments
Name	Sensor Name	N/A	
Time	Time	[s]	
Type	Sensor Type (informational only – not displayed in the GUI)	N/A	
Status	Status (1=Unknown, 2=Faulty, 3=Below-min, 4=nominal, 5=above-max)	N/A	
Temperature	Temperature	[°C]	

Statistics available for Solaris

SUN_DEAMON metric

Statistics	Description	Unit	Comments
Name	DaemonName	N/A	
Time	Time	[s]	
CpuLoad	CPU average Cpu load during last sample interval	[%]	
MemSize	MEM memory allocated by this process	[MB]	

SUN_DEVICE metric

Statistics	Description	Unit	Comments
Name	Block device name	N/A	
Time	Time	[s]	
DevRate	DEV transfer rate	[1/s]	
DevKB	DEV throughput	[kB/s]	
DevRdRate	DEV read rate	[1/s]	
DevRdKB	DEV read throughput	[kB/s]	
DevWrRate	DEV write rate	[1/s]	
DevWrKB	DEV write throughput	[kB/s]	

SUN_FILESYS metric

Statistics	Description	Unit	Comments
Name	File System name	N/A	
Time	Time	[s]	
Size	Total Size	[GB]	
Used	Used Size	[GB]	
Free	Free Size	[GB]	

SUN_IP metric

Statistics	Description	Unit	Comments
Name	Protocoll Name	N/A	
Time	Time	[s]	
iDGTot	Total Datagram rate (in & out)	[1/s]	
iDGRcv	Datagram receive rate	[1/s]	
iDGXmt	Datagram transmit rate	[1/s]	
iRcvDeliver	IN Rate of successful datagram delivery to IP-stack	[1/s]	
iRcvForward	IN Forwarding datagram rate	[1/s]	
iRcvHdrErr	IN IP-Header error rate	[1/s]	
iRcvAddrErr	IN Address error rate	[1/s]	
iRcvUnkProt	IN Unknown protocol rate	[1/s]	
iRcvDiscard	IN Datagram discard rate	[1/s]	
iXmtNoRout	OUT Datagram discard rate	[1/s]	
iXmtDiscard	OUT Datagram discard due to no route	[1/s]	

SUN_NIC metric

Statistics	Description	Unit	Comments
Interface	Interface Name	N/A	
Time	Time	[s]	
OctTot	TOT total number of octets (rcv & tmx)	[kB/s]	
UCasts	TOT total number of subnet-unicast packets	[1/s]	
Discards	TOT total number of packets discarded	[1/s]	
ErrTot	TOT total errors on interface	[1/s]	
InOct	RCV octets received	[kB/s]	
InUCast	RCV number of subnet-unicast packets	[1/s]	
InDiscards	RCV inbound msg discarded without error	[1/s]	
InErr	RCV inbound msg with error	[1/s]	
OutOct	TMX octets transmitted	[kB/s]	
OutUCast	TMX number of subnet-unicast packets requests	[1/s]	
OutDiscards	TMX outbound msg discarded without error	[1/s]	
OutErr	TMX outbound msg with error	[1/s]	
OutQue	TMX output packet queue	[#]	

SUN_PROCESS metric

Statistics	Description	Unit	Comments

PrcName	Process Name	N/A
Pid	Process ID	N/A
Time	Time	[s]
RunPath	Path from which the SW was loaded	N/A
RunParameter	Parameters applied to this process when initially loaded	N/A
CpuLoad	CPU average Cpu load during last sample interval	[%]
MemSize	MEM memory allocated by this process	[MB]

SUN_SYSTEM metric

Statistics	Description	Unit	Comments
Node	OS name	N/A	
Time	Time	[s]	
CpuIdle	CPU Cpu utilization idle state	[%]	
CpuLoad	CPU Cpu utilization	[%]	
CpuUser	CPU Cpu utilization user state	[%]	
CpuSystem	CPU Cpu utilization system state	[%]	
CpuKernel	CPU Cpu utilization kernel state	[%]	
CpuWait	CPU Cpu utilization wait state	[%]	
PhyMemSize	MEM physical memory available	[MB]	
PhyMemFree	MEM physical memory currently free	[MB]	
PhyMemUse	MEM physical memory currently used	[MB]	
SwapTot	MEM total swap space	[MB]	
SwapFree	MEM free swap space	[MB]	
SwapUsed	MEM used swap space	[MB]	
PrcTot	total number of process	[#]	
UserSessTot	PRC total number of user sessions	[#]	
DevRate	DEV systemwide device transfer rate	[1/s]	
DevKB	DEV systemwide device throughput	[kB/s]	
DevRdRate	DEV systemwide device read rate	[1/s]	
DevRdKB	DEV systemwide device read throughput	[kB/s]	
DevWrRate	DEV systemwide device write rate	[1/s]	
DevWrKB	DEV systemwide device write throughput	[kB/s]	

SUN_TCP metric

Statistics	Description	Unit	Comments
Name	Protocoll Name	N/A	
Time	Time	[s]	
iSegTot	Total Segment rate (in & out)	[1/s]	

iSegRcv	Segement receive rate	[1/s]
iSegXmt	Segment transmit rate	[1/s]
iSegReXmt	Segment re-transmit rate	[1/s]
iCurrEstab	Current established TCP connections	[#]
iActOpen	Connect request rate (CLOSED -> SYN-SENT)	[1/s]
iPassOpen	Connection accept rate (LISTEN -> SYN-RCVD)	[1/s]
iAttFail	Connection request & accept fail rate	[1/s]
iEstabRest	Connection close rate	[1/s]

Statistics available for Linux

LINUX_DEAMON metric

Statistics	Description	Unit	Comments
PrcName	DaemonName	N/A	
Time	Time	[s]	
CpuLoad	CPU average Cpu load during last sample interval	[%]	
MemSize	MEM memory allocated by this process	[MB]	

LINUX_FILESYS metric

Statistics	Description	Unit	Comments
Name	File System name	N/A	
Time	Time	[s]	
Size	Total Size	[GB]	
Used	Used Size	[GB]	
Free	Free Size	[GB]	

LINUX_IP metric

Statistics	Description	Unit	Comments
Name	Protocoll Name	N/A	
Time	Time	[s]	
iDGTot	Total Datagram rate (in & out)	[1/s]	
iDGRcv	Datagram receive rate	[1/s]	
iDGXmt	Datagram transmit rate	[1/s]	
iRcvDeliver	IN Rate of successful datagram delivery to IP-stack	[1/s]	
iRcvForward	IN Forwarding datagram rate	[1/s]	
iRcvHdrErr	IN IP-Header error rate	[1/s]	
iRcvAddrErr	IN Address error rate	[1/s]	
iRcvUnkProt	IN Unknown protocol rate	[1/s]	
iRcvDiscard	IN Datagram discard rate	[1/s]	
iXmtNoRout	OUT Datagram discard rate	[1/s]	
iXmtDiscard	OUT Datagram discard due to no route	[1/s]	

LINUX_NIC metric

Statistics	Description	Unit	Comments

Interface	Interface Name	N/A
Time	Time	[s]
OctTot	TOT total number of octets (rcv & tmx)	[kB/s]
UCasts	TOT total number of subnet-unicast packets	[1/s]
Discards	TOT total number of packets discarded	[1/s]
ErrTot	TOT total errors on interface	[1/s]
InOct	RCV octets received	[kB/s]
InUCast	RCV number of subnet-unicast packets	[1/s]
InDiscards	RCV inbound msg discarded without error	[1/s]
InErr	RCV inbound msg with error	[1/s]
OutOct	TMX octets transmitted	[kB/s]
OutUCast	TMX number of subnet-unicast packets requests	[1/s]
OutDiscards	TMX outbound msg discarded without error	[1/s]
OutErr	TMX outbound msg with error	[1/s]
OutQue	TMX output packet queue	[#]

LINUX_PROCESS metric

Statistics	Description	Unit	Comments
PrcName	Process Name	N/A	
Pid	Process ID	N/A	
Time	Time	[s]	
RunPath	Path from which the SW was loaded	N/A	
RunParameter	Parameters applied to this process when initially loaded	N/A	
CpuLoad	CPU average Cpu load during last sample interval	[%]	
MemSize	MEM memory allocated by this process	[MB]	

LINUX_SYSTEM metric

Statistics	Description	Unit	Comments
Node	OS name	N/A	
Time	Time	[s]	
CpuIdle	CPU Cpu utilization idle state	[%]	
CpuLoad	CPU Cpu utilization	[%]	
CpuUser	CPU Cpu utilization user state	[%]	
CpuSystem	CPU Cpu utilization system state	[%]	
CpuKernel	CPU Cpu utilization kernel state	[%]	
CpuWait	CPU Cpu utilization wait state	[%]	
PhyMemSize	MEM physical memory available	[MB]	
PhyMemFree	MEM physical memory currently free	[MB]	
PhyMemUse	MEM physical memory currently used	[MB]	
SwapTot	MEM total swap space	[MB]	
SwapFree	MEM free swap space	[MB]	
SwapUsed	MEM used swap space	[MB]	
PrcTot	total number of process	[#]	
UserSessTot	PRC total number of user sessions	[#]	

LINUX_TCP metric

Statistics	Description	Unit	Comments
Name	Protocoll Name	N/A	
Time	Time	[s]	
iSegTot	Total Segment rate (in & out)	[1/s]	
iSegRcv	Segement receive rate	[1/s]	
iSegXmt	Segment transmit rate	[1/s]	
iSegReXmt	Segment re-transmit rate	[1/s]	
iCurrEstab	Current established TCP connections	[#]	
iActOpen	Connect request rate (CLOSED -> SYN-SENT)	[1/s]	
iPassOpen	Connection accept rate (LISTEN -> SYN-RCVD)	[1/s]	
iAttFail	Connection request & accept fail rate	[1/s]	
iEstabRest	Connection close rate	[1/s]	

Statistics available for RDB

CACHE metric

This metric provides cumulative information for all row caches in the database.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iLatchRqs	CACHE latch (atomic data structure modifications) rate	[1/s]	
iLatchRetry	CACHE latch (atomic data structure modifications) retry rate	[1/s]	
iCachSearch	CACHE row cache search rate for particular DBKEY	[1/s]	
iCachWS	CACHE rate a particular row was found in process working set	[1/s]	
iCacheGbl	CACHE rate a particular row was found in the global row cache	[1/s]	
iCacheBig	CACHE rows too big to fit into the specified cache buffer	[1/s]	
iCacheInsert	CACHE row cache insert rate	[1/s]	
iCacheInsBig	CACHE row inserts too big to fit into the spec. cache buffer	[1/s]	
iCacheInsFull	CACHE row cache full rate (no disk flush - all rows modified)	[1/s]	
iCacheInsColl	CACHE row cache hash table collision rate when inserting rows	[1/s]	
iVLMRqs	CACHE VLM request rate	[1/s]	
iVLMWinTurn	CACHE VLM window turn rate	[1/s]	
iCacheDirtSkip	CACHE entries not replaced due to modified data content	[1/s]	
iCacheUseSkip	CACHE entries not replaced due to (other) process reference	[1/s]	
iCacheHashMiss	CACHE row cache hash table misses (overfows)	[1/s]	
iCacheUnmark	CACHE row flush rate to disk	[1/s]	

CACHE.UNMARK metric

This metric provides row cache "unmark" statistics which describe how rows in the row caches are written back to disk.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iGrant	CACHE row lock (latch) grant rate	[1/s]	
iStall	CACHE row lock (latch) stall rate	[1/s]	
iDead	CACHE row lock (latch) deadlock rate	[1/s]	
iRel	CACHE row lock (latch) release rate	[1/s]	
iStallTime	CACHE Avg. row lock (latch) stall time within last sample	[ms/s]	

INDEX.HASH metric

This metricprovidesstatistics aboutthe update and retrieval activity of a database's hashed indexes. It indicates the total number of key insertions and deletions, the number of scans that were opened, and for retrievals (successful fetches), the total number of nodes (either bucket fragments or duplicate nodes) that were fetched.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iInsert	HASH hash key insert rate	[1/s]	
iInsDup	HASH duplicate has key update rate	[1/s]	
iDelete	HASH hash key delete rate	[1/s]	
iDelDup	HASH duplicate key deletion rate	[1/s]	
iScan	HASH hash index scan rate (retrieval & update)	[1/s]	
iFetch	HASH hash node fetch rate on successful scans	[1/s]	
iFetchBuckFrag	HASH fragmeneted bucket fetch rate on successful scans	[1/s]	
iFetchDupNode	HASH duplicate node fetch rate on successful scans	[1/s]	

INDEX.INSERTION metric

This metricprovidesstatistics about the update activity of a database's sorted indexes during insertions; that is, when you store or modify an index key field or when you use the SQL CREATE INDEX statement on a table. This screen also indicates in which type of index node the insertions occur and displays node creations by node type. By examining this screen, you can monitor how a database balances its sorted indexes after insertions into the database.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iIdxIns	INDEX overall index insertion rate (all index nodes)	[1/s]	
iIdxInsRoot	INDEX index insertion rate into root-level index nodes	[1/s]	
iIdxInsLeaf	INDEX index insertion rate into bottom level index nodes	[1/s]	
iIdxInsDup	INDEX duplicate index insertion rate	[1/s]	
iNodeCre	INDEX overall index node creation rate	[1/s]	
iNodeCreRootSp	INDEX root-level index node split rate due to overflow	[1/s]	
iNodeCreLeaf	INDEX bottom level index node create rate	[1/s]	
iNodeCreDup	INDEX duplicate node creation rate	[1/s]	

INDEX.REMOVAL metric

This metricprovidesstatistics about the update activity of a database's sorted indexes when you perform any removal operation; that is, erase,alter, or modify an index key field or drop or delete an index. This screen indicates from which type of index node the removals occur. It also shows node deletions by node type. This screenlets you monitor how a database balances its sorted indexes when nodes are removed from the indexes.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iIdxRem	INDEX overall index removal rate (all index nodes)	[1/s]	
iIdxRemRoot	INDEX index removal rate into root-level index nodes	[1/s]	
iIdxRemLeaf	INDEX index removal rate into bottom level index nodes	[1/s]	
iIdxRemDup	INDEX duplicate index removal rate	[1/s]	
iNodeDel	INDEX overall index node delete rate (due to SQL DROP INDEX)	[1/s]	
iNodeDelLeaf	INDEX bottom level index node delete rate	[1/s]	
iNodeDelDup	INDEX duplicate node delete rate	[1/s]	

INDEX.RETRIEVAL metric

This metricprovidesstatistics of how much retrieval activity is taking place in a database's sorted indexes. Oracle Rdb often uses direct index lookups and index scans to access records in the database. This screen monitors these operations as well as the number of index nodes fetched.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iVerbSuc	TX Verbs executions succees rate	[1/s]	
iVerbRollBack	TX Verbs roll back rate	[1/s]	
iNodeFetch	INDEX overall index node fetch rate	[1/s]	
iNodeLeaf	INDEX bottom-level index node fetch rate	[1/s]	
iNodeDup	INDEX duplicate index node fetch rate	[1/s]	
iIdxLookup	INDEX direct single-key retrievals performed on database indexes	[1/s]	
iIdxScan	INDEX database index scan rate	[1/s]	
iIdxScanPrim	INDEX Avg. unique keys found per Index scan	[1/n]	
iIdxScanDup	INDEX Avg. duplicate keys found per Index scan	[1/n]	

IO.ASYNCH_IO metric

This metric provides information concerning asynchronous reads and writes to the database files.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iIO	PIO Asynchronous read/write disk I/Os for data/SPAM pages	[1/s]	
iStall	PIO stall count rate due to I/O completion waits (read&write)	[1/s]	
iDataRdReq	PIO Asynchronous data page read reqquest rate	[1/s]	
iDataRdIO	PIO Asynchronous disk read I/O rate to get data pages	[1/s]	
iSPAMRdReq	PIO Asynchronous SPAM page read reqquest rate	[1/s]	
iSPAMRdIO	PIO Asynchronous disk read I/O rate to get SPAM pages	[1/s]	
iReadStall	PIO stall count rate due to read I/O completion waits	[1/s]	
iWriteIO	PIO Asynchronous data/SPAM page write I/O rate	[1/s]	
iWriteStall	PIO stall count rate due to write I/O completion waits	[1/s]	

IO.FETCH metric

This metric provides statistics on how data and SPAM page requests are handled.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
PIOType	Type	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iFetchRead	PIO data page request rate with read privileges only	[1/s]	
iFetchUpd	PIO data page request rate with update & read privileges	[1/s]	
iASok	PIO data page req. found in users's allocate set (AS) ok	[1/s]	
iASLckNeeded	PIO data page req. found in AS that requires add. locking	[1/s]	
iASOldVers	PIO data page req. in AS with old version (disk re-read)	[1/s]	
iGBLckNeeded	PIO data page req. found in global buffer pool (lock req.)	[1/s]	
iGBOldVer	PIO data page req. GB with old version (disk re-read)	[1/s]	
iGBTransfer	PIO data page transfer rate between process via memory	[1/s]	
iPgIO	PIO data pages not found in GB and read from disk	[1/s]	
iPgSynth	PIO data pages 'synthesized' and not read from disk	[1/s]	

IO.FILE metric

This metric allows you to display I/O statistics for each file in the database.

- AIJ file
- ACE file
- RUJ file
- Root file
- All Data/snap files

With the exception of the all data/snap files element, each element in the metric shows the I/O activity for a specific database file. The all data/snap files screen shows a summary of I/O activity for all data and snapshot files.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
LockType	Lock Type	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iIOTot	FIO Total file I/O rate	[1/s]	
iIOTotBlk	FIO Total file I/O - Avg. block transfer per I/O	[Blk]	
iIOTotComp	FIO Total file I/O - Avg. I/O completion time per I/O	[ms]	
iSyRd	FIO Sync read file I/O rate	[1/s]	
iSyRdBlk	FIO Sync read file I/O - Avg. block transfer per I/O	[Blk]	
iSyRdComp	FIO Sync read file I/O - Avg. I/O completion time per I/O	[ms]	
iSyWr	FIO Sync write file I/O rate	[1/s]	
iSyWrBlk	FIO Sync write file I/O - Avg. block transfer per I/O	[Blk]	
iSyWrComp	FIO Sync write file I/O - Avg. I/O completion time per I/O	[ms]	
iExt	FIO file extends total	[#]	
iExtBlk	FIO file extend - Avg. block transfer per I/O	[Blk]	
iExtComp	FIO file extend - Avg. I/O completion time per I/O	[ms]	
iAsyRd	FIO Async read file I/O rate	[1/s]	
iAsyRdBlk	FIO Async read file I/O - Avg. block transfer per I/O	[Blk]	
iAsyRdComp	FIO Async read file I/O - Avg. I/O completion time per I/O	[ms]	
iAsyWr	FIO Async write file I/O rate	[1/s]	
iAsyWrBlk	FIO Async write file I/O - Avg. block transfer per I/O	[Blk]	
iAsyWrComp	FIO Async write file I/O - Avg. I/O completion time per I/O	[ms]	

IO.PREFETCH metric

This metric provides statistics on asynchronous data and SPAM page pre-fetching.Asynchronous pre-fetching includes both traditional asynchronous pre-fetching ("APF") when pages are "known" to be needed (as during a sequential scan of a storage area) and detected asynchronous pre-fetching ("DAPF") when a pattern of sequential page access is noticed and pages are automatically pre-fetched in anticipation.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
PIOType	Type	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iAPFSuc	PIO APF initiated buffer fetch success rate	[1/s]	
iAPFFail	PIO APF initiated buffer fetch failure rate (lock conflict)	[1/s]	
iAPFUtil	PIO buffer access rate already fetched by APF	[1/s]	
iAPFWaste	PIO buffers (fetched by APF) removed without being accessed	[1/s]	
iDAPFSuc	PIO DAPF initiated buffer fetch success rate	[1/s]	
iDAPFFail	PIO DAPF initiated buffer fetch failure rate (lock conflict)	[1/s]	
iDAPFUtil	PIO buffer access rate already fetched by DAPF	[1/s]	
iDAPFWaste	PIO buffers (fetched by DAPF) removed without being accessed	[1/s]	

IO.STALL_IO metric

This metric shows a summary of I/O stall activities.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iROOTRead	PIO Avg. ROOT file read time within last sample	[ms/s]	
iROOTWrite	PIO Avg. ROOT file write time	[ms/s]	
iDataRead	PIO Avg. data and snapshot file read time within last sample	[ms/s]	
iDataWrite	PIO Avg. data and snapshot file write time within last sample	[ms/s]	
iDataExt	PIO Avg. data and snapshot file extend time within last sample	[ms/s]	
iRUJRead	PIO Avg. RUJ file read time within last sample	[ms/s]	
iRUJWrite	PIO Avg. RUJ file write time within last sample	[ms/s]	
iRUJExt	PIO Avg. RUJ file extend time within last sample	[ms/s]	
iAIJRead	PIO Avg. AIJ file read time within last sample	[ms/s]	
iAIJWrite	PIO Avg. AIJ file write time within last sample	[ms/s]	
iAIJExt	PIO Avg. AIJ file extend time within last sample	[ms/s]	
iAIJHiber	PIO Avg. AIJ I/O completion hibernate time within last sample	[ms/s]	

JOURNAL.2PC metric

This metric provides information about distributed transaction performance and how it differs from non-distributed transaction performance.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
i2PCTx	2PC distributed transaction rate (comitted & rolled back)	[1/s]	
iTxTotTime	TX Avg. transaction duration (all transactions)	[ms]	
iTxRegTime	TX Avg. regular transaction duration	[ms]	
iTx2PCTime	2PC Avg. distributed transaction duration	[ms]	
iTxPrep	TX Avg. time spent in transaction resultion phase within last sample	[ms/s]	
i2PCResolve	2PC Processes failed during prepare state (recoverd by DBR)	[1/s]	
i2PCCom	2PC Comitted distributed Transaction rate	[1/s]	
i2PCRollB	2PC rollbacked distributed Transaction rate	[1/s]	

JOURNAL.AIJ metric

This metric provides statistics about both logical and physical after-image journaling activity.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iTotWrites	AIJ overall AIJ-I/O write rate	[1/s]	
iDataWrites	AIJ data write-I/O (data records only) rate	[1/s]	
iCtrlWrites	AIJ Ctrl-I/O rate (open/commit/rollback/checpoint records)	[1/s]	
iExtends	AIJ number of file extends	[#]	
iSwitches	AIJ number of AIJ file switches	[#]	
iRecords	AIJ record write rate	[1/s]	
iTotReads	AIJ overal AIJ-I/O read rate	[1/s]	

JOURNAL.ALS metric

This metric contains information specific to the operation of the AIJ Log server process.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iTotWrites	AIJ overall AIJ-I/O write rate	[1/s]	
iDataWrites	AIJ data write-I/O (data records only) rate	[1/s]	
iCtrlWrites	AIJ Ctrl-I/O rate (open/commit/rollback/checpoint records)	[1/s]	
iExtends	AIJ number of file extends	[#]	
iSwitches	AIJ number of AIJ file switches	[#]	
iWriteTime	AIJ Avg. AIJ file write time within last sample	[ms/s]	
iExtTime	AIJ Avg. AIJ file extend time within last sample	[ms/s]	
iHiberCnt	ALS AIJ log server hibernate rate waiting for work	[1/s]	
iHiberTime	ALS Avg. process hibernate time while ALS processes their requests	[ms/s]	
iGrpCom	ALS group commit rate performed by AIJ log server	[1/s]	
iARBFmt	ALS ARBs (AIJ request block) formatted by ALS process	[1/s]	
iARBBckGFmt	ALS ARBs background formatted during asy. I/O to AIJ journal	[1/s]	

JOURNAL.DBR metric

This metric identifies various recovery phases and shows information on how long each phase took to complete. The Recovery Statistics metric is useful for identifying an excessive number of abnormal process failures. In addition, the screen is useful for determining the proper database attribute and parameter settings to maximize runtime performance minimize recovery downtime.

Note that this screen provides global information on all failed process recoveries, not on individual process recoveries.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iPrcFail	DBR abnormally process term. rate (database recovery required)	[1/s]	
iDBFreeze	DBR database freeze time within last sample interval	[ms/s]	
iTXRedoCnt	DBR transaction REDO count	[1/s]	
iTXRedoTime	DBR Avg. transaction REDO time within last sample interval	[ms/s]	
iTXUndoCnt	DBR transaction UNDO count	[1/s]	
iTXUndoTime	DBR Avg. transaction UNDO time within last sample interval	[ms/s]	
iTxNoUndoCnt	DBR transaction count not needed to be 'undone'	[1/s]	
iTxCom	DBR transactions committed during transaction UNDO phase	[1/s]	
iTxRollb	DBR transactions rolled back during transaction UNDO phase	[1/s]	
iTxNoResolve	DBRTX not needed to be committed or rolled back during UNDO	[1/s]	
iAIJRecTime	DBR time to recover in-progress AIJ flushes within last sample	[ms/s]	
iBGRecTime	DBR time to recover global buffer info within last sample	[ms/s]	
iCacheRecTime	DBR time to recover row cache within last sample interval	[ms/s]	

JOURNAL.RUJ metric

This metric contains summary information for all active update transactions on the current node.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iTotWrites	RUJ overall RUJ-I/O write rate	[1/s]	
iDataWrites	RUJ data write-I/O (data records only) rate	[1/s]	
iCtrlWrites	RUJ Ctrl-I/O rate (open/commit/rollback/checpoint records)	[1/s]	
iExtends	RUJ number of file extends	[#]	
iRecords	RUJ record write rate	[1/s]	
iTotReads	AIJ overal RUJ-I/O read rate	[1/s]	

LOCK.TYPE metric

This metric contains informationabout database locking activities of different lock types:

- Total database locks
  The statistics in this screen are the totals for all types of database locks.
- Database physical area locks
- Database page locks
  Page locks are used to manage the database page buffer pool.
- Database record locks
  Record locks are used to maintain the logical consistency of the database. Allrecord locks in the adjustable lock granularity tree are included here.
- Database Sequence block (SEQBLK) locks
  The SEQBLK locks maintain global transaction sequence numbers or transaction and commit sequence numbers and control COMMIT and ROLLBACK operations.
- Database file identification (FILID) locks
  The FILID locks are used to maintain consistent end-of- file information for .rdb, .rda, and .snp database files.
- Database Transaction block (TSNBLK) locks
  The TSNBLK locks are used to control the COMMIT and ROLLBACK operations on each OpenVMS cluster node. TSNBLK locks are also used to control SQL SET TRANSACTION statements for read-only transactions.
- Database Run-time user process block (RTUPB) locks
  The RTUPB lock is used to maintain a consistent list of the users who are attached to the database.
- Database ACTIVE user bitmap locks
  The ACTIVE lock is used to maintain a consistent list (in bit map form) of the users who are attached to the database.
- Database MEMBIT node bitmap locks
  The MEMBIT lock is used to maintain a consistent list (in bit map form) of the nodes on which the database is currently accessed.
- After-image journal locks (AIJ)
  The AIJ locks are used to control reading and writing to the .aij file. One global AIJ lock maintains current end-of-file information. In addition, one local AIJ lock on each OpenVMScluster node manages the global AIJ buffers on that node.
- Database Snapshot locks
  The snapshot locks are used to manage the allocation of snapshot pages to users who are updating the database. Snapshot locks are only used if snapshots are enabled for a storage area.
- Database Freeze locks

The freeze lock is used to suspend database activity while a database recovery process is running.

- Database Quiet-point locks
  The quiet-point lock suspends starting new transactions while the AIJ despooler is trying to finish despooling the contents of the primary .aij file when you use the RMU Backup command. The quiet-point lock also suspends starting new transactions during the startup of an online RMU Backup command.
- Database logical area locks
  Logical area locks are obtained when Oracle Rdb readies tables. Lock carryover can help reduce the number of logical area locks.
- Database NOWAIT transaction locks
- Database client information (CLIENT) locks
  The CLIENT locks are used to provide serialized access to the database metadata stored in the system stored. The CLIENT locks are also used to serialize operations such as creating tables and indices.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
LockType	Lock Type	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iENQReq	LCK ENQ request (new ENQ & CVT) rate total	[1/s]	
iDEQRel	LCK DEQ request rate total	[1/s]	
iCVThigh	LCK Conversion (CVT) to higher Lock mode rate	[1/s]	
iCVTlow	LCK Conversion (CVT) to lower Lock mode rate	[1/s]	
iLckOutStand	LCK outstanding lock rate	[1/s]	
iENQNotQue	LCK new ENQ request not queued due to lock conflict	[1/s]	
iENQNotQue	LCK new ENQ request not queued due to lock conflict	[1/s]	
iENQStall	LCK new ENQ request stalled due to lock conflict	[1/s]	
iENQTimOut	LCK new ENQ request time out	[1/s]	
iENQDeadLock	LCK new ENQ deadlock count	[1/s]	
iCVTNotQue	LCK CVT to higher Lock mode not queued due to lock conflict	[1/s]	
iCVTStall	LCK CVT to higher Lock mode stalled due to lock conflict	[1/s]	
iCVTDeadLck	LCK CVT to higher Lock mode deadlock count	[1/s]	
iCVTTimOut	LCK CVT to higher Lock mode time out	[1/s]	
iBlkAst	LCK Blocking AST delivery to RDB	[1/s]	
iWaitTime	LCK Avg. lock stall time spent by all users within last sample	[ms/s]	

LOGNAM metric

This metric provides statistics on database dashboard updates and logical name translation.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iLogNamTrans	LOGNAM logical name translation rate	[1/s]	
iLogDefault	LOGNAM default assignment rate to logical names	[1/s]	
iDashUpd	LOGNAM user notification rate due to database dashborad update	[1/s]	

OBJECT.TYPE metric

This metric provides statistics about database objects of different types:

- KROOT object
  The KROOT object contains the database control information that describes all of the other database objects.
- FILID object
  The FILID object contains the storage area information.
- SEQBLK object
  TheSEQBLK object contains the information on the allocation of sequence numbers such as transaction sequence numbers (TSNs) and commit sequence numbers (CSNs).
- TSNBLK object
  TheTSNBLK object contains the information on the last committed transaction.

  The number of TSNBLK objects is a function of the maximum number of users in the database; there is one TSNBLK object for every 28 database users (rounded up). For example, a database containing a maximum of 512 users would contain 19 TSNBLK objects.
- AIJDB object
  The AIJDB object contains the AIJ control information.
- AIJFB object
  The AIJFB object contains the AIJ journal information.
- RTUPB object
  The RTUPB object contains information on active users.
- ACTIVE object
  The ACTIVE object contains information on active transactions.
- CPT object
  The CPT object contains information on the corrupt page table.
- RCACHE object
  The RCACHE object contains the row cache information.
- CLIENT object
  The CLIENT object contains client-specific information.
- CLTSEQ object
  The CLTSEQ object contains the client sequence information.
- UTILITY object
  The UTILITY object contains information used by the RMU utility.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
LockType	Lock Type	[N/A]	

Time	Time	[s]
iTxTotal	TX Transaction rate	[1/s]
iFetchShr	OBJ Objects fetched shared	[1/s]
iFetchEx	OBJ Objects fetched for exclusive access	[1/s]
iRefresh	OBJ Objects info staled - refetched from root file	[1/s]
iModify	OBJ Objects info modified	[1/s]
iWritten	OBJ Objects info rewritten to database root file	[1/s]
iReleased	OBJ Objects released	[1/s]

RECORD metric

This metric provides a summary of data row activity for storage areas in the database. Data row activity includes modification (marked), retrieval (fetch), store, or erase operations.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iRecMark	REC records marked (records modiefied / erased)	[1/s]	
iRecFetch	REC records fetched (incl. snapshot records)	[1/s]	
iRecFetchFrag	REC record fragments fetched (incl. snapshot records)	[1/s]	
iRecStor	REC records stored in database	[1/s]	
iRecStoFrag	REC records stored fragmented in database	[1/s]	
iPgChk	REC page check rate (to store records)	[1/s]	
iPgChkInBuf	REC page check rate of pages already in buffer	[1/s]	
iPgChkDis	REC page check discard rate due to low free space	[1/s]	
iRecErase	REC records erased from database	[1/s]	
iRecEraseFrag	REC fragmented records erased from database	[1/s]	

SNAPSHOT metric

This metric provides statistics about snapshot activity for both update and read-only transactions. The "TX record retrieve record", "TX lines fetch rate", and "SNAP snapshot page fetch rate" statistics relate to read-only transactions, and the last five statistics are relevant for update transactions.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iTxRO	TX READ ONLY Transaction rate	[1/s]	
iRec	TX records retrieve rate	[1/s]	
iLine	TX lines fetch rate	[1/s]	
iPage	SNAP snapshot page fetch rate	[1/s]	
iStore	SNAP record store rate in snapshot file	[1/s]	
iPgUsed	SNAP 'page owned by other transaction' fetch fail rate to store rec.	[1/s]	
iPgFull	SNAP 'page full' fetch fail rate to store rec.	[1/s]	
iPgLck	SNAP 'page lock confict' fetch fail rate to store rec.	[1/s]	
iPgExtend	SNAP snapshot file extensions	[#]	

STALLS metric

This metric provides statistics about process stall rates for every stall class. Stall classes available:

- Miscellaneous
- Records
- Pages
- Tables
- Storage areas
- Database root-file
- Recovery journals
- User transactions
- Hot standby
- Database

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iTotCnt	STALL overall (all stall classes) process stall count	[1/s]	
iTotTime	STALL avg. overall stall time within last sample (all classes)	[ms/s]	
iMiscCnt	STALL generic process stall rate	[1/s]	
iMiscTime	STALL avg. generic process stall time within last sample	[ms/s]	
iRecCnt	STALL record-related process stall rate	[1/s]	
iRecTime	STALL avg. record-related stall time within last sample	[ms/s]	
iPageCnt	STALL page-related process stall rate	[1/s]	
iPageTime	STALL avg. page-related stall time within last sample	[ms/s]	
iTblCnt	STALL table-related process stall rate	[1/s]	
iTblTime	STALL avg. table-related stall time within last sample	[ms/s]	
iAreaCnt	STALL area-related process stall rate	[1/s]	
iAreaTime	STALL avg. area-related stall time within last sample	[ms/s]	
iFileCnt	STALL root-file-related process stall rate	[1/s]	
iFileTime	STALL avg. root-file-related stall time within last sample	[ms/s]	
iJourCnt	STALL journal-related process stall rate	[1/s]	
iJourTime	STALL avg. journal-related stall time within last sample	[ms/s]	
iTransCnt	STALL user transaction-related process stall rate	[1/s]	
iTransTime	STALL avg.user transaction-related stall time within last sample	[ms/s]	
iHotStbCnt	STALL hot-standby-related stall rate	[1/s]	
iHotStbTime	STALL avg.hot-standby-related stall time within last sample	[ms/s]	
iDBCnt	STALL database-related process stall rate	[1/s]	
iDBTime	STALL avg. database-related stall time within last sample	[ms/s]	

TRANS metric

This metric summarizes database transaction activity and indicates transaction and verb execution rates.

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate	[1/s]	
iTxTime	TX Avg. Transaction duration	[ms]	
iTxRO	TX READ ONLY Transaction rate	[1/s]	
iTxROTime	TX Avg. READ ONLY Transaction duration	[ms]	
iTxRW	TX READ WRITE Transaction rate	[1/s]	
iTxRWTime	TX Avg. READ WRITE Transaction duration	[ms]	
iTxCom	TX Transaction rate committed	[1/s]	
iTxRollBack	TX Transaction rate rolled back	[1/s]	
iTxRBTime	TX Avg. Transaction rollback duration	[ms]	
iTxPrep	TX Transactions prepared	[1/s]	
iVerbSuc	TX Verbs executions succees rate	[1/s]	
iVerbRollBack	TX Verbs roll back rate	[1/s]	
iVerbRBTime	TX Avg. Verb rollback duration	[ms]	
iCkPt	TX Checkpoint rate	[1/s]	
iCkPtTime	TX Avg. Checkpoint operation duration	[ms]	

TRANS.HISTOGRAM metric

This metric provides a histogram of transaction durations during a sample interval. Such a transaction duration histogram is available for:

- All transactions
- RO transactions
- RW transactions only

Statistics	Description	Unit	Comments
DBName	Database Name	[N/A]	
TransType	Transaction type	[N/A]	
Time	Time	[s]	
iTxTotal	TX Transaction rate during last sample interval	[1/s]	
TxTime_01	TX rate – trans. duration range 1 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_02	TX rate – trans. duration range 2 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_03	TX rate – trans. duration range 3 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_04	TX rate – trans. duration range 4 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_05	TX rate – trans. duration range 5 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_06	TX rate – trans. duration range 6 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_07	TX rate – trans. duration range 7 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_08	TX rate – trans. duration range 8 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_09	TX rate – trans. duration range 9 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_10	TX rate – trans. duration range 10 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_11	TX rate – trans. duration range 11 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_12	TX rate – trans. duration range 12 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_13	TX rate – trans. duration range 13 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use

TxTime_14	TX rate – trans. duration range 14 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_15	TX rate – trans. duration range 15 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_16	TX rate – trans. duration range 16 (RDB version specific)	1/s#]	Time range covered depends on the RDB Version in use
TxTime_17	TX rate – trans. duration range 17 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_18	TX rate – trans. duration range 18 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_19	TX rate – trans. duration range 19 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_20	TX rate – trans. duration range 20 (RDB version specific)	[1/s]	Time range covered depends on the RDB Version in use
TxTime_21	TX rate – trans. duration range 21 (RDB version specific)	1/s#]	Time range covered depends on the RDB Version in use