

Chapter 1



Fidelia NetVigil™ Overview

1.1 Introduction

Fidelia NetVigil™ is a breakthrough solution that provides integrated fault and performance monitoring of applications, networks, systems and user defined data sources. This versatile data gathering combination enables you to rapidly create business containers – personalized views showing the real-time performance of business services, derived from the performance of your underlying IT infrastructure. This technology has established Fidelia as a leader in the business services management arena.

The object-oriented components of NetVigil's architecture are capable of automatically determining relationships between problems in the infrastructure and business services. With its open, easily extensible APIs and data feeds, NetVigil can monitor any device or application that can be instrumented. Powerful Data Gathering Engines (DGEs), each with its own database, automatically discover problems and establish baselines and thresholds for monitored applications, networks, and systems. Business containers can be created on the fly and can represent a geographic location, a business unit, or a revenue-generating service. Containers can share elements with other containers.

NetVigil features an intuitive point-and-click browser-based user interface that seamlessly integrates fault and performance data in a unified view. NetVigil's capabilities include a sophisticated "delegated user authority," which lets you distribute responsibilities for personal infrastructure slices to other users in your organization. In addition, NetVigil is highly scalable, easily scaling to support tens of thousands of geographically dispersed networks, systems and applications.

Real-time Fault and Performance

NetVigil can run tests against your applications, databases, network equipment or servers and indicate faults when the test fails or crosses a preset threshold (such as for database transaction rates, web server response times, disk space, bandwidth utilization, etc.). It can also parse for patterns in log files, receive SNMP traps, and generate alarms when a pattern matches.

In addition to detecting faults in real-time, NetVigil stores the collected data using real-time progressive aggregation techniques to store the performance data for extended periods of time (up to several years) with modest database size requirements (around 1GB per year of data). This historical data is then used for trend analysis, capacity planning reports and baseline reports based on statistical analysis of past data.

NetVigil uses a unique distributed database and processing model to generate reports in real time from large volumes of historical data which is not available using traditional data warehousing techniques.

Flexible Service Container Views

NetVigil allows users to create flexible “containers” of applications, devices or tests in order to see the end-to-end performance of a “service”. For example, a “Payroll service” might have a database, a printer and a payroll application all connected via a network router. This feature allows the user to create a “Payroll Service Container” and monitor all underlying components of that service in a single view. The status of the containers is updated in real-time based on the status of its components. Additionally, these containers can be nested and one can determine service impact using the container reports. Containers can be created automatically based on rules, and you can set the status of the container using rule logic also (for redundant IT elements, etc.)

Delegated Authority User Model

NetVigil has a unique delegated user model which allows multiple departments in an enterprise to each have their own ‘virtual management system’ without being able to see each other’s data, while allowing certain ‘administrators’ to have read-only or read-write access to multiple departments. As an example, the network department, the server group, the database group and the application group can each have their own private accounts on the system, while allowing the

HelpDesk to have a read-only view across all the departments and the Operation Center to have a read-write view across all or some of the departments.

In a Service Provider environment, this feature can be used to offer managed services to customers very easily.

Event manager for Operation Centers

The Netvigil Event Manager allows powerful & distributed filtering of syslogs, windows events, SNMP traps and then acknowledge, suppress or delete these events using the Event Manager Console. Ideal for Network Operation Center (NOC) environments, the web based interface can be accessed by multiple operators in a distributed datacenter environment.

Notification Engine

NetVigil has a extensible action & notification engine that features automatic escalation of problems over time, time of day based notification and allows suppression of “dependent” alerts so as to prevent alarm floods. If an eCommerce service is down because an unreachable database because of an intermediate switch failing, the system can send out a single notification about the switch instead of sending a flood of alarms for everything that is unreachable. New actions can be added easily using the plugin framework.

Extensible APIs

NetVigil has very powerful APIs which allow access to all components of the software. Users familiar with Perl or C can start using the API very quickly due to its familiar commands and interface. These APIs allow connecting to other legacy products or custom applications with minimal effort.

Highly Distributed and Scalable

The NetVigil architecture is horizontally scalable and uses distributed databases and parallel processing to deliver real-time fault and performance reports. Additional reporting engines and data collectors can be added to the system as needed to scale to very large networks, and the BVE layer automatically presents a unified view across all the distributed data collectors. The entire software can be run on a single machine for a small environment, or scale to hundreds of thousands of IT elements by deploying on multiple distributed servers.

The distributed DGE model allows NetVigil to handle multiple NAT networks or firewall-protected LANs that may exist in large enterprises.

Easy to Use

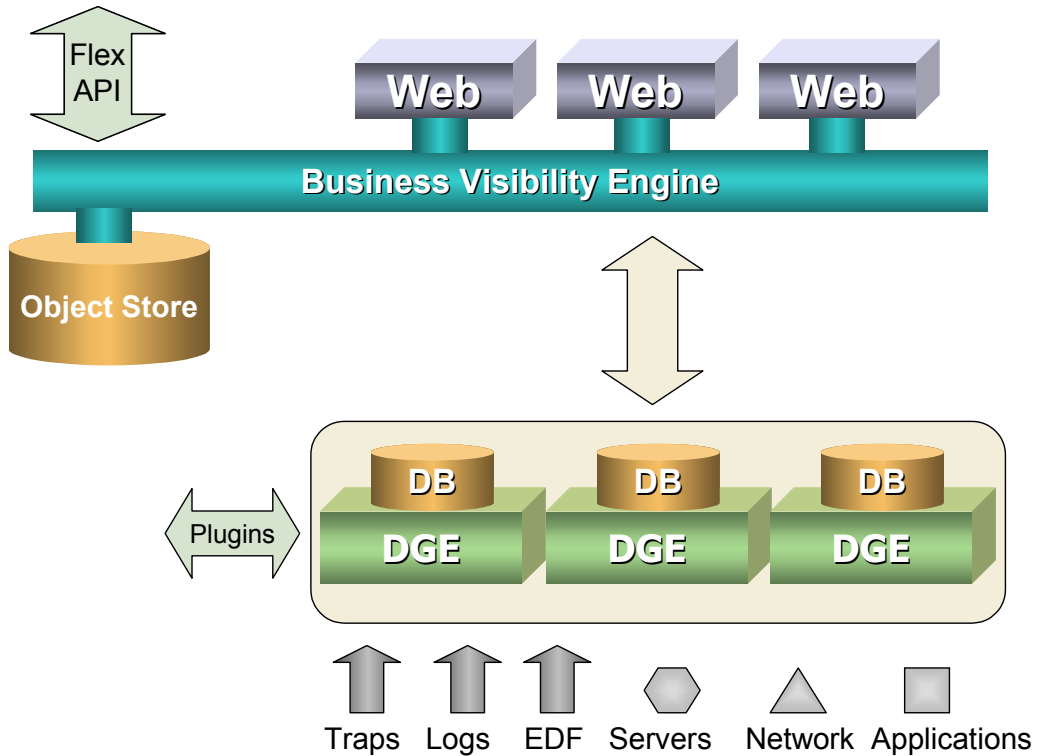
NetVigil was designed with a Web UI as the primary interface making it extremely simple to use. You can access all of its functionality by using any web browser on your desktop. The product is easy to install by downloading over the web and running a discovery on specified subnets. The system automatically discovers devices, applications and servers on your network, and starts monitoring them with pre-defined thresholds (which can be modified or automatically updated by using the baselining feature).

1.2 NetVigil Architecture

NetVigil uses a 3-tier architecture (See figure below) consisting of the Business Visibility Engine (BVE) WebApp, the BVE ObjectStore, and one or more Data Gathering Engines (DGEs). All configuration and authentication information is stored in the BVE ObjectStore (embedded Object Oriented database).

For enterprises, all of the NetVigil hardware may exist at one central location or be geographically distributed to accommodate multiple offices or divisions. One DGE can monitor all the devices at a single location, including servers, routers, switches, applications, and network appliances. The DGE measures and stores aggregated performance data locally, and forwards only events or alarms to the BVE components. For a data center environment, it is recommended

that you employ a DGE at each data center location and set up the provisioning and authentication database at a central location (e.g., your NOC).



NetVigil System Components

The NetVigil system comprises three main components. In a large setup, each component should run on a different host.

1. The **BVE ObjectStore** is an embedded object oriented database that stores all configuration information. This includes metadata related to user authentication, devices, tests, thresholds for test results, action profiles and other key information. The BVE FlexAPI, which allows access to the BVE for provisioning and results, also runs on this machine.

2. The **BVE WebApp** provides the web based user interface into NetVigil. It correlates the data from multiple DGEs, and allows end users to look at the real-time status of their devices, add new devices and actions, run reports, etc. using a simple web browser. It handles the distributed databases and distributed processing seamlessly while generating the real-time reports and graphs. It is possible to have more than one BVE WebApp for load sharing, in which case any load balancing hardware could be utilized to load-share all access across the WebApps.
3. The **Data Gathering Engines (DGE)** do the actual polling of data, receive SNMP traps, generate alarms based on thresholds, and do the aggregation of data in real-time. DGEs should be located as close as possible to the devices being monitored to reduce wide area network traffic. These DGEs can be geographically dispersed or you can even have multiple DGEs in the same location to distribute the load across different physical servers. When you have multiple DGEs in the same location, the system automatically provisions new devices onto the DGE with a lower number of devices.

Although the three components are designed to be on different machines, NetVigil's flexible design permits two or more components to be set up on a single physical machine and operate without any problem.

DGEs schedule and perform tests, archive and aggregate data, and trigger notifications and actions. Effective management of historical information is accomplished by setting bounds on storage that prevent it from growing to unmanageable limits. Historical records are aggregated and stored for over a year by default. Historical alarm and event data (i.e. changes in severity level) are retained without aggregation and only aged by user selection.

During the initial installation, existing department or device records (or a subset thereof) can be imported into the NetVigil provisioning database using the BVE FlexAPI, to deliver quick installation and configuration. The system comes with reasonable default thresholds for all tests, which can be automatically updated using the 'baselining' feature after running for a few days. If firewalls are used within the data center, then access through the firewalls must be provided to enable monitoring of the devices behind them. If the number of devices behind a firewall is significant, the DGE can alternatively be connected

to a port behind the firewall. In addition, if you are using Network Address Translation (NAT) or private address space, the IP address being used must be unique within the data center.

1.3 Overview of System Operation

Each component of NetVigil operates independently to provide a high level of scalability and fault tolerance. When a DGE is started, it looks in its `dge.xml` file for its unique name, which must match the name that was specified when the DGE was created (for details on creating DGEs, see Chapter 10, “DGE Management”). The DGE connects to the BVE ObjectStore (specified in the `netvigil.xml` file) and downloads the entire configuration associated with that unique name, including tests, thresholds and actions.

From then on, the DGE operates fairly independently. It performs tests, generates events when thresholds are crossed, and triggers the corresponding notifications. The data collected by each DGE is stored in a local SQL database on the DGE itself.

Any configuration changes made in the BVE ObjectStore (via the BVE API or the webApp) is instantly pushed out to the appropriate DGE.

A user logs into the web application, which gets the list of devices that the user has permission to view from the config database. The web application then connects directly to the distributed DGEs and gets the real-time status of the services or devices. When the user needs a report, the Web application fetches the data using parallel queries from the distributed DGEs and generates the reports in real-time.

Log files and SNMP traps are handled by the Input Stream Monitor (ISM) and the trap daemon respectively, and then matched against user-defined regular expressions that define severities and corresponding actions/notifications. These text based messages can be viewed in the Message Window in the BVE WebApp.

1.4 NetVigil Terms

A **DGE** is the Data Gathering Engine that polls devices for the various tests such as CPU, bandwidth, etc., and aggregates the data that it gathers. Generally, a DGE is physically near the devices that it

monitors. A DGE can typically monitor approximately 500 - 1,500 devices (see Section 10.1.4, “Disk Space Requirements for DGE Aggregation” on page 147 for more precise sizing algorithms).

A **DGE Location** is a collection of one or more DGEs that are automatically load balanced for provisioned tests. The DGEs within a single DGE Location are usually located in the same physical region, but they can be separate in some special situations. When you provision a device, you assign it to a DGE Location, not to an individual DGE. If there is more than one DGE at that location, NetVigil automatically assigns the device to the least loaded DGE.

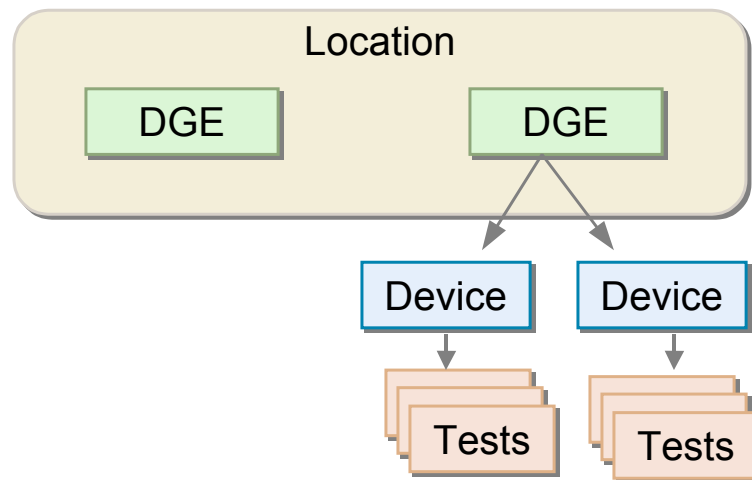


Figure 1.1 Relationship between Locations, DGEs, Devices, Tests

NetVigil monitors the performance of your network and application systems, and their underlying components. These systems and components, referred to as **devices**, can be routers, switches, servers, databases, networks, or applications.

Tests are measures of device functioning and are used to monitor your devices. NetVigil reports the status of each test. **Test status** (shown on the **Device Details** page) is the current status category (ok, warning, critical, unknown, unreachable, suspended, unknown or unconfigured) for a test. **Device status** (shown on the **Status Summary** page) is the worst current test status for a device.

NetVigil uses boundaries called **thresholds** to determine a test's status. A **threshold** is the outer limit of acceptable performance on a variable such as utilization, packet loss, etc. An **event** occurs whenever a test result crosses a threshold. These events form the basis for reporting via logs or graphs. Events also trigger NetVigil's actions.

An **action** is an activity that is automatically triggered by an event. Actions can be designed to take place as soon as a single event occurs, or after the same event occurs repeatedly. For instance, an E-mail notification can be sent whenever a test crosses the warning threshold, or it can be sent after a test has crossed the warning threshold five consecutive times. Certain action types are built into NetVigil (e.g., E-mail, pager, external scripts). In addition, the plugin framework allows you to add new types of actions as needed. Plugins are described in detail in Chapter 26, "Plugin Actions".

A **Department** is an entity that allows creation and assignment of devices. Devices are owned by Departments. Departments relate to a customer organization or department, and can have multiple end-user logins.

End-users have unique NetVigil login names with either read-only or read-write privileges to create and modify devices, tests, or actions within a Department.

An **Administrator** is a special type of user with the ability to: create and modify Departments and the devices, tests, and actions owned by those Departments; populate default test thresholds; and establish service level permissions and limits for Departments.

To simplify privilege and limit management for end-users, Departments are associated with **User-Classes**. Similarly, all administrative users are associated with **Admin-Classes**. The **superuser** then creates the privileges matrix associating the Admin-Class with one or more User-Classes. This privilege matrix describes what members of the Admin-Class can do to members of the User-Class.

