

Chapter 1



Fidelia Helix™ Overview

1.1 Introduction

Fidelia Helix™ is a flexible monitoring solution that provides integrated fault and performance monitoring of applications, networks, systems and other user defined data sources. The “virtual container” technology enables you to rapidly create service containers and report on them in real-time.

Leveraging the latest object-oriented technologies in the industry, Helix provides one of the most flexible architectures in the industry today. Its real-time reporting engine can calculate statistics, generate graphs and tables on demand within seconds. With its open, easily extensible APIs and data feeds, Helix can monitor any device or application that can be instrumented. More importantly, Helix provides a seamless transition path for smaller enterprises to its flagship NetVigil product which offers unmatched scalability in the industry today.

Helix features an intuitive point-and-click, browser-based user interface that drills down to illustrate infrastructure fault and performance with red-amber-green traffic lights. The product was designed with a web interface as the primary interface, minimizing training requirements and maximizing ROI.

Real-time Fault and Performance

Helix 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.).

In addition to detecting real-time faults, Helix stores the collected data for extended periods of time (several years) using real-time progressive aggregation. It then performs trend analysis and generates performance reports from this historical data. Helix uses a relational database to store all data and employs progressive aggregation techniques to store the performance data for extended periods of time (up to several years) with modest database size requirements.

Flexible Service Container Views

Helix allows users to create flexible “containers” of tests in order to see the performance of an end to end “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.

Extensible Plug-in Architecture

Helix has very powerful plug-in architecture which allows adding new monitors and actions very easily.

Easy to Use

Helix was designed with a Web UI as the primary interface from the ground up, and as such is 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 (even downloadable over the web), automatically discovers devices, applications and tests on your network, and starts monitoring them with pre-defined thresholds.

1.2 Helix Architecture

Helix uses a 3-tier software architecture comprised of the Business Visibility Engine (BVE) WebApp, the BVE ObjectStore, and a Data Gathering Engine (DGE). All configuration and authentication information is stored in the BVE ObjectStore (embedded Object Oriented database).

The **BVE WebApp** provides the web based user interface into Helix. It correlates the data from the DGE database, 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 also has the real-time reporting engine for generating the real-time reports and graphs.

The **Data Gathering Engine (DGE)** does the actual polling of data, generates alarms based on thresholds, and does the aggregation of data in real-time. It schedules and perform tests, archives and aggregates data, and triggers notifications and actions.

Smart Data Management

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.

1.3 Helix Terms

Helix 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. Helix 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.

Helix 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 Helix'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 Helix (e.g., E-mail, pager, external scripts). Plugins are described in detail in Chapter 11, "Plugin Actions".

End-users have unique Helix 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 users, do the initial network discovery and do other initial setups.