Data Solutions from Bradmark

Bradmark Surveillance 4.7 Concepts Guide

Notice

Disclaimer and Copyright

Contents

List of Figures

Chapter 1

Landscape

Topics:

Bradmark Surveillance is navigated by visiting servers, or SAMs, and drilling down to agents, entities, etc. Then windows can be opened and actions performed. You can directly log into a SAM, or reach the SAM by seamlessly browsing from a list of SAMs at a central location (the *CAS*).

You can also jump directly to a window from an alert in the Alerts Dashboard.

SAM

A *SAM* represents a server that is being monitored.

Clicking on a SAM in the Surveillance tree provides access to entities (connections to databases, etc.) and product-related options.

Administrators of Surveillance go to the SAM to configure security and permissions functionality, such as SAM user accounts to log into Surveillance, and role-based access control (RBAC) and fine-grained permissions to filter views or data based on the audience.

By default, a SAM name is the hostname of the server. However, SAM names cannot contain hyphens, so these will be converted to underscores during installation.

Related concepts

CAS on page 10
A *Central Alerter Server (CAS)* provides a central location to configure alerts and to seamlessly browse Satellite (SAT) SAMs.

SAT on page 11
If a CAS is being used, then any SAM that registers with the CAS is a *Satellite (SAT)* SAM.

Entity

An *entity* defines a connection to the database, operating system (OS), or network that will be monitored.

Clicking on an entity in the Surveillance tree provides access to real-time views, flashback, recent history, rules, and stores.

Entities can be enabled or disabled, which controls whether rules and stores are started; disabling an entity will prevent alerts and historical data (but still allows real-time views, as well as recent history on data already collected). In a failover scenario, cluster failover actions can be defined in your clustering solution to automatically disable entities on the old server and enable entities (to the same databases) on the new server.

OS entities are added automatically to the Surveillance tree, but database entities must be created. In web client, you can create the entity using the most popular options, and there are notes provided on how to ensure that the database is configured properly. In the Windows client, you can use Entity Wizard, which will ensure that the entity can connect to the database instance and that it has the necessary permissions to monitor the database, as well as configure various monitoring options.

Domains

SAMs are organized into *domains*.

This is typically the suffix of the fully-qualified domain name (FQDN) of the servers, but it can also be customized.

A CAS can host multiple domains.

CAS

A *Central Alerter Server (CAS)* provides a central location to configure alerts and to seamlessly browse Satellite (SAT) SAMs.

The CAS is like a central hub; it allows you to browse and monitor many SATs. A typical workflow is to log into the CAS, then browse to the SAT with the entity (database) that you want to see. In this way, you don't need to log into each SAT individually. You can also open multiple views (tabs in the web client), where each view comes from the same SAT or different SATs, allowing you to view metric data from multiple servers at once.

When you look at alerts on the CAS (Alerts tab in web client), you see alerts from many SATs. It provides a global dashboard of problems everywhere. Just like browsing SATs, the alerts that you see are governed by permissions that have been granted to your login.

Multiple CAS SAMs can be used to create a multi-tier landscape, as well. For example, each region could have a CAS, and a global CAS can sit above that level. Alerts can propagate from SATs to regional CAS to global CAS. Alert management at higher levels automatically propagate to lower levels in the hierarchy.

Related concepts

SAT on page 11
If a CAS is being used, then any SAM that registers with the CAS is a *Satellite (SAT)* SAM.

SAM on page 10
A *SAM* represents a server that is being monitored.

Entity on page 10
An *entity* defines a connection to the database, operating system (OS), or network that will be monitored.

SAT

If a CAS is being used, then any SAM that registers with the CAS is a *Satellite (SAT)* SAM.

The SAT SAM (server) is where monitoring data and alerts are generated. It has the entities (databases, OS) which are being monitored.

To view metric data for a database, you log into the CAS and then browse to the SAT. Then you view monitoring data by entity. The list of monitoring views which are available is dictated by which permissions have been granted to your login and by properties of the entity, such as database version.

For administrators of Surveillance: the cassetup utility is used to register a SAT with its CAS. This will automatically enroll the SAT in Hearbeat monitoring, to alert on whether the SAT server is up or down. Additionally, a FORWARD event action on a global event handler (*.*.*) needs to be configured on the SAT to forward all alerts onto to the CAS.

Related concepts

CAS on page 10
A *Central Alerter Server (CAS)* provides a central location to configure alerts and to seamlessly browse Satellite (SAT) SAMs.

SAM on page 10
A *SAM* represents a server that is being monitored.

Entity on page 10
An *entity* defines a connection to the database, operating system (OS), or network that will be monitored.

Chapter 2

Alerts

Topics:

An *alert* provides notification of a condition that you are interested in monitoring.

In most cases, this condition is a problem or potential problem that you want to avoid. For example, you may want to get alerts when a server becomes unavailable.

Bradmark Surveillance automates the monitoring process through rules analysis and event generation. From these events, the system determines the actions that need to be taken. The events and the actions performed in response to those events compromise the alerts.

This process begins by defining and applying data rules in the Analyzer Agent. When the agent is started, it loads its rule set, attaches to the appropriate data collections, and begins a continuous process of evaluating rules. If a rule has been satisfied (for example, something noteworthy has occurred) it generates an event that is sent to the local Alerter. The Alerter receives the event, and determines which, if any, Event Handler is appropriate for handling the event. An incident is then created in the Active Incident Queue and performs the actions defined in the Event Handler. When all actions have been completed, the incident is closed.

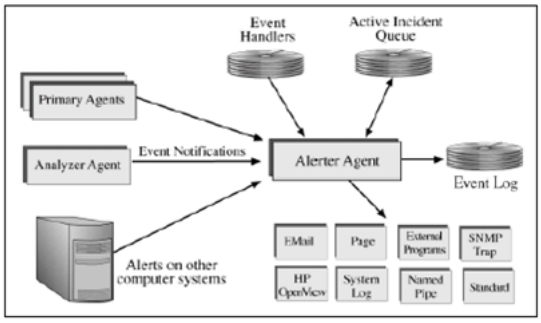

Figure 1: Alerting Overview

For example, when defining a rule that informs that the Average Number of New Sessions has exceeded 25, the name of the event is generated by the rule, AVG_NEW_SESSIONS_ EXCEEDED. This rule is then applied, or turned on, for a database. When the database encounters an Average Number of New Sessions that exceeds 25, an event with the Event ID “AVG_NEW_SESSIONS_EXCEEDED” is generated and sent to the Alerter Agent.

In order to handle the event AVG_NEW_SESSION_EXCEEDED, the Alerter Agent will require an event handler to be defined. The event handler might have two actions defined: E-mail and Page the DBA.

If an event handler has no actions defined, then only an incident is opened.

Open incidents are what drive the Alerts Dashboard. The Alerts Dashboard reflects a summary of the open incidents for a given server or entity. For example, if a critical event for low free space arrives, then an incident is opened, and that entity may show as red in the Alerts Dashboard. But once the low free space condition is cleared, then the incident is closed, and the entity may show as green in the Alerts Dashboard.

On a Satellite Server, the only event action defined may be a FORWARD action, to forward all alerts to the Central Alerter Server (CAS). Then, on the CAS, there may be event actions centrally defined, in addition to reflecting open incidents on its Alert Dashboard.

Event

An *event* indicates that an alert may be generated.

Each event has an *event ID*, which uniquely identifies the event and is referenced in event handlers and other places. For example, a **NORAD_DOWN** event indicates that the Bradmark Surveillance software on a server is not responding, while a **SERVER_DOWN** event indicates that the server itself has stopped responding to network ping requests, as well.

Rules generate events, and events trigger actions in event handlers.

Rules

A *rule* creates an event when certain criteria are met on data in a collection.

For example, if a disk is 80% full, then a rule might generate a warning event.

The Analyzer Agent has the ability to formulate complex rules using criteria involving multiple variables from entity collection definitions. In addition, rule definitions can be used as a rule template that can be applied over and over again to several databases with different thresholds, refresh rates, and other characteristics.

Standard Rules

A *standard rule* is a rule defined to activate automatically for every entity.

Each entity can be individually enabled or disabled for standard rules, to include or exclude them from this process.

Event Handlers

An *event handler* defines how events should be tracked and what actions, if any, should be taken when an event occurs.

The Alerter Agent is responsible for evaluating events, determining the appropriate actions, and monitoring the status of the actions. By default, the Alerter logs events to the Event Log and discards them. An event handler must be configured to tell the Alerter how to process events.

Event Actions

An *event action* is an action performed as part of an event handler, in response to one or more events.

For example, an email event action may provide notification that free space is too low.

Most Event Handlers will contain a list of one or more actions to be performed; however, actions are not required. An action is comprised of action-specific components and components common to all actions.

Action Templates

An *action template* is one of the action types (for example, SNMP trap, e-mail, external, etc.) that has been defined in advance for use within Event Handlers as an action.

Action Templates provide two primary benefits:

- Action Templates aid in managing actions that are common to many event handlers.
- Action Templates provide a starting point for defining an action within an event handler.

Chapter 3

Historical

Topics:

Bradmark Surveillance provides short-term and long-term historical metric data, in addition to real-time metrics.

Data is initially stored in a local cache repository, or stores, which are a set of proprietory flat files. This data is used for short-term history, such as for Flashback or Time Slicing.

Then that data can be optionally sent to a Central Repository database on an interval (i.e. every 30 minutes), for long-term reporting.

Stores

A *store* is a table kept in the local cache repository, which is a set of proprietory flat files.

It consists of a rolling set of partitions, and retains data for a certain number of days.

Standard Stores

A *standard store* is a store defined to activate automatically for every entity.

Each entity can be individually enabled or disabled for standard stores, to include or exclude them from this process.

Recent History

Recent historical data is pulled from the stores in the local cache repository, local to each server.

Flashback

Flashback allows you to take the user interface, used for real-time monitoring, back to a point-in-time of the recent past (i.e. yesterday at 10:00, or last week at this time).

You can jump directly to a time in the past, then navigate forward or backward by a minute, or several minutes, or even 10 seconds. All windows are updated to the point-in-time, allowing you to correlate metrics from multiple windows.

Time Slicing

Time Slicing is a report on recent history that shows top resource usage over a period of time.

For example, you can see the longest-running SQL from 13:00 to 14:00 yesterday. It enables quick, ad-hoc analysis of recent history.

There are two types of Time Slicing reports:

- **By Time Range**. Easily investigate resource consumption within a specific time range, i.e. Top SQL between 15:00 and 16:00 yesterday, showing elapsed time, CPU time, and more.
- **Top N Summary**. From Top SQL By Hour to Top Processes By Day, a graphical summary view presents the largest resource consumers in a time period.

Long-term History

Long-term historical data is kept in the Central Repository.

It can be exported to this database on an interval (i.e. every 30 minutes). Data can be kept for months or years, and reduction facilities are provided.

Central Repository

The Central Repository is a database (i.e. ASE, SQL Server) that stores long-term historical data, i.e. for long-term trend analysis.

Data can be exported from stores to the Central Repository database on an interval (i.e. every 30 minutes). Web Reporting can be used to query the Central Repository.

Data reduction facilities are available to summarize data at different age levels. The repository must be initialized before first use.

Reports

Web Reporting provides long-term history reports from the web client.

Chapter 4

Real-time

Topics:

Bradmark Surveillance provides *real-time monitoring* through windows, or views, in clients.

Every refresh interval (i.e. 30 seconds), the window metrics are updated automatically, providing the latest information on the current state of the database or OS.

Factory Collections

A *factory collection* is a collection in Bradmark Surveillance that is shipped with the product.

Factory collections cover a wide variety of areas, from collecting metrics on running SQL to space usage, from CPU usage to disk statistics, and much more.

User-defined Collections

A *user-defined collection* (UDC) is a custom collection that extends the product by collecting data using a SQL statement, OS command, or other source.

The UDC becomes a first-class object, so rules, stores, and windows can be created against the new collection, just like factory collections.

A UDC is defined by a text file under a subdirectory of $NSM_HOME/cfg/Collections.

Chapter 5

Architecture

Topics:

Surveillance can operate as an agentless or agent-based product.

A web-based user interface (web client) provides access, along with other tools.

As an *agentless* product, Surveillance is installed on a machine and can monitor databases remotely. The Surveillance Server agents just run on the same machine as the client or web tier. This is how Surveillance Express Edition works, and is a great way to trial the product.

Surveillance is more commonly deployed as an *agent-based* product. The Surveillance Server agents, or SAM, reside on each server that runs a database. This provides the most reliable, efficient, and extensive monitoring possible. You can also have databases being monitored remotely, creating a hybrid agent-based and agentless deployment.

Architecture Overview

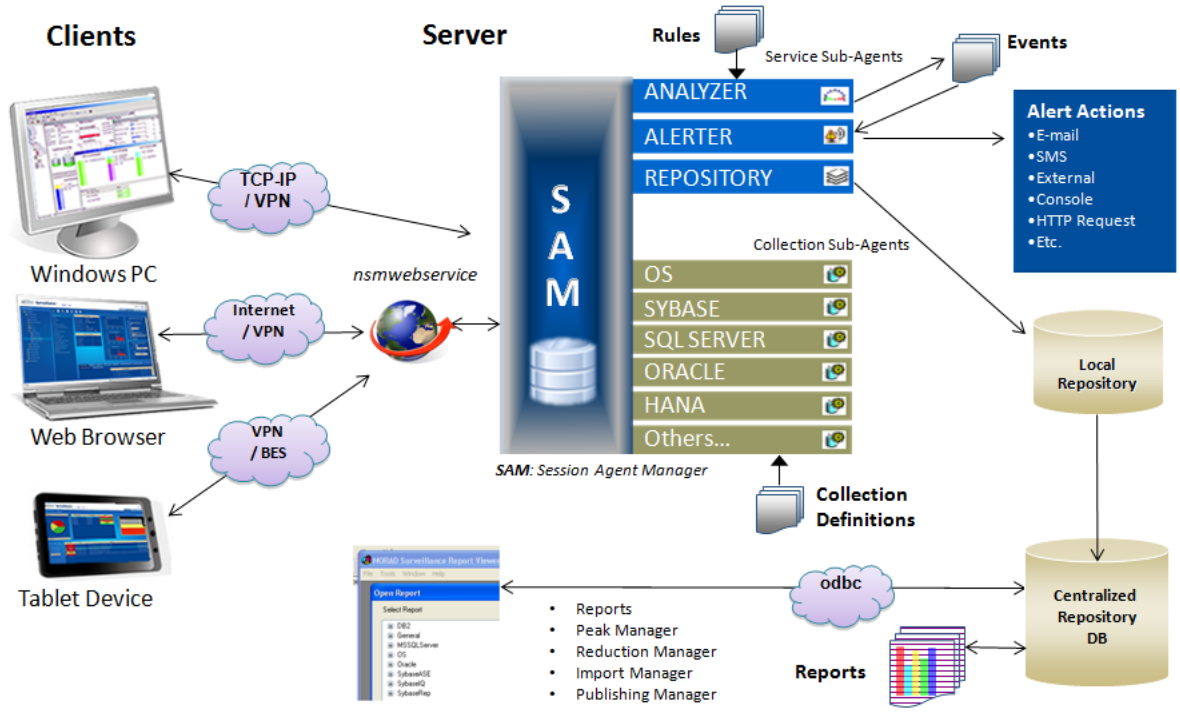

Figure 2: Surveillance Architecture Overview

SAM

A *SAM* represents a server that is being monitored.

Clicking on a SAM in the Surveillance tree provides access to entities (connections to databases, etc.) and product-related options.

The Session Agent Manager (SAM) is the heart of Surveillance Server. It sits between the client and server agents, directing traffic and delivering messages. The SAM determines which agent should perform work requested from the client. It provides security and permissions functionality, such as SAM user accounts to log into Surveillance, and role-based access control (RBAC) and fine-grained permissions to filter views or data based on the audience.

A SAM automatically starts most agents, which run continuously, but it may start and stop some agents on demand. The SAM process is called `dbgsam` (UNIX and Linux) or `dbgsamnt.exe` (Windows). Most agents have names that begin with "dbg" or "nsm".

The SAM provides the listener on certain network interfaces and ports, waiting for logins from client users or other SAMs. Each connection creates a virtual session (vsession), and these are multiplexed over physical sessions (psessions). For example, SAM A can open a physical session to SAM B, and then multiple virtual sessions can flow over this physical session. Most of this session management happens automatically; when you log into the CAS and browse to a SAT, the two SAMs handle all of the details to provide a seamless browsing experience.

By default, a SAM name is the hostname of the server. However, SAM names cannot contain hyphens, and these will get converted to underscores during installation.

Agents

Surveillance Server is a set of programs used to collect data and automate analysis of the database servers.

The core technology behind individual monitoring components is a generic, high-performance, configurable monitoring and object brokering technology. This technology was developed specifically for monitoring database performance in a distributed heterogeneous platform and vendor environment.

Surveillance Server is the program that performs all the work. It runs a wide variety of platforms. Surveillance Server collects data, performs automated data analysis, and acts on that analysis by issuing alerts. It also performs any work that a client requests. For example, if the user wants to display a real-time sessions window, Surveillance Server would perform the collection and pass the information to the client. Surveillance Server contains multiple agents designed to perform specific tasks on the server.

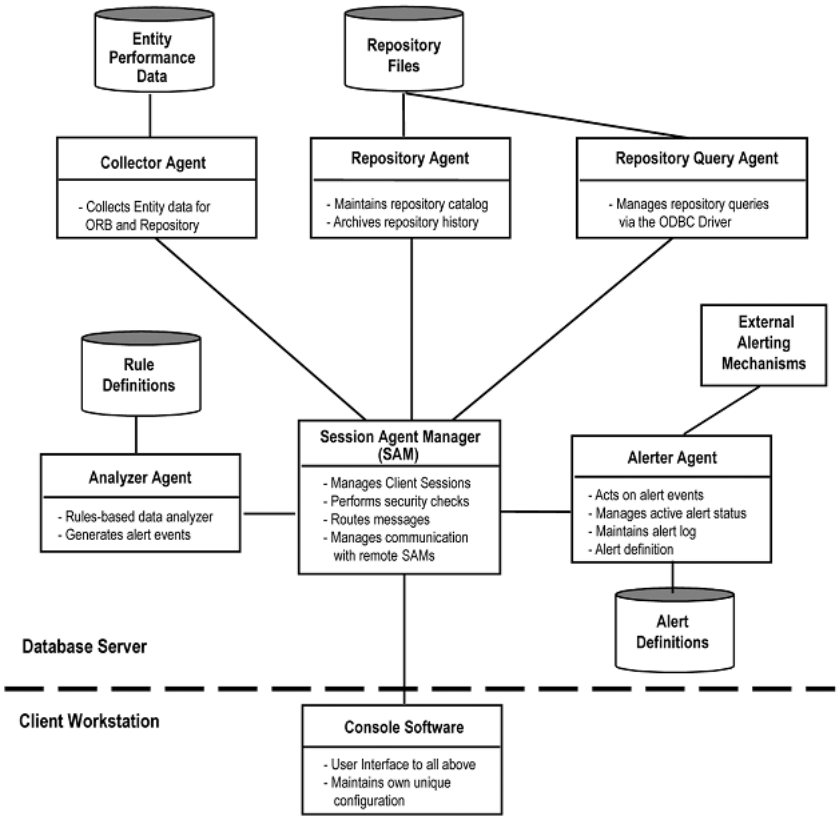

Figure 3: Agents Overview

Collector Agents

Configured with a set of SQL scripts or other technologies, known as Factory Collections, a *Collector Agent* collects information from a database, the OS, or network.

The client makes a request to the collector to retrieve information, one time or at specified intervals. The collector then manages information retrieval. The collector is designed for efficiency. If there are simultaneous requests for the same information, the collector queries the data only once.

User-defined Collections (UDCs) extend monitoring capabilities to application data and data integrity. User-defined Collections provide the flexibility to add custom performance metrics, as well as monitor application-specific rules. After the data has been collected, a built-in derive function can be utilized to calculate deltas and values. The User-defined Collections can also be used to define rules and send alerts based on the data returned from the new collection, and all collection results can be stored in the repository.

Web Tier

The web tier of Bradmark Surveillance is provided through the `nsmwebservice` process.

Clients

Bradmark Surveillance can be accessed by a variety of client programs, from a full UI to command-line tools.

Web Client

The *web client* is a web-based user interface to Bradmark Surveillance.

It requires Flash player 10.0 or later. TLS, for transport encryption, is supported.

Console

The *console* (or console.exe) is a command-line text client for Bradmark Surveillance.

This tool is for advanced operations or scripting. There is also a Console alert action available.

Windows Client

The Bradmark Surveillance Windows client is the legacy Windows GUI tool.

The *thin client* can be extracted to any directory, followed by running `setup-thinclient.cmd`.

Entity Wizard can be executed from the Windows client for advanced entity creation.

Index

D