SQL Sentry Components and Architecture

Last Modified on 25 May 2021

Overview

SQL Sentry Components

Note: Due to product name changes between versions and branding, you may have *SentryOne*, *SQL Sentry*, or a combination of these names in your installed SQL Sentry components.

SQL Sentry consists of the following components:

Component	Purpose
SQL Sentry Client	Provides a thin client interface for viewing and managing collected information. Associated application services and processes: • **SentryOne Client.exe**
SQL Sentry Monitoring Service	Collects event metadata and historical information for the monitored device(s). Associated application services and processes: • **SQLSentryServer** / **SentryOne.App.MonitoringService.exe** • **SQLSentryServer** / **SQLSentry.App.DiskSynchronizer** (**Note:** prior to version 2021.8, it was **SentryOne.App.DiskSynchronizer.exe**)
SQL Sentry Database	A SQL Server database that stores event metadata and historical information collected by the SQL Sentry monitoring service. **Note:** The default name for this database is *SQLSentry*. Other versions may have used *SentryOne* as the default. Some setups may use custom names, such

Component	Purpose
	as *SQLSentry-US* and *SQLSentry-UK*.
SQL Sentry Controller	When using EPI, this service controls the SQL Sentry services, allowing them to be managed remotely. It must exist on all machines where a monitoring service, portal service, or client is installed for the EPI version of SentryOne. Associated application services and processes: • **SentryOneController / SentryOne.App.ClientBootstrapper.exe** • **SentryOneController / SentryOne.App.ServiceBootstrapper.exe**
SQL Sentry Portal	If using the self-hosted version of SQL Sentry Portal, this service runs on the machine where the portal is installed. Associated application services and processes: • **SentryOneMonitorPortal / SentryOne.Monitor.WebClient.Web.exe**
Miscellaneous processes & services (things that may need to be added to a safelist in your security software, if applicable)	• **SentryOne Report Viewer / SsrsViewer.exe** • **SQL Sentry Scheme Handler / SqlSentrySchemeHandler.exe** • **Startup Wizard / SqlSentry.StartupWizard.exe** • **Service Configuration Utility / SentryOne.App.MonitoringService.Configuration.exe**

Notes

One SQL Sentry monitoring service is typically required for every 50 to 100 monitored targets. Install multiple monitoring services for scalability, redundancy, or to collect information from remote sites. Normally a SQL Sentry client is installed on each DBA's workstation.

Note:

- See the Implementation Examples section of the Recommendations article for additional information on the number of SQL Sentry monitoring services.
- All SQL Sentry monitoring services and clients connect to the same SQL Sentry database.

Important:

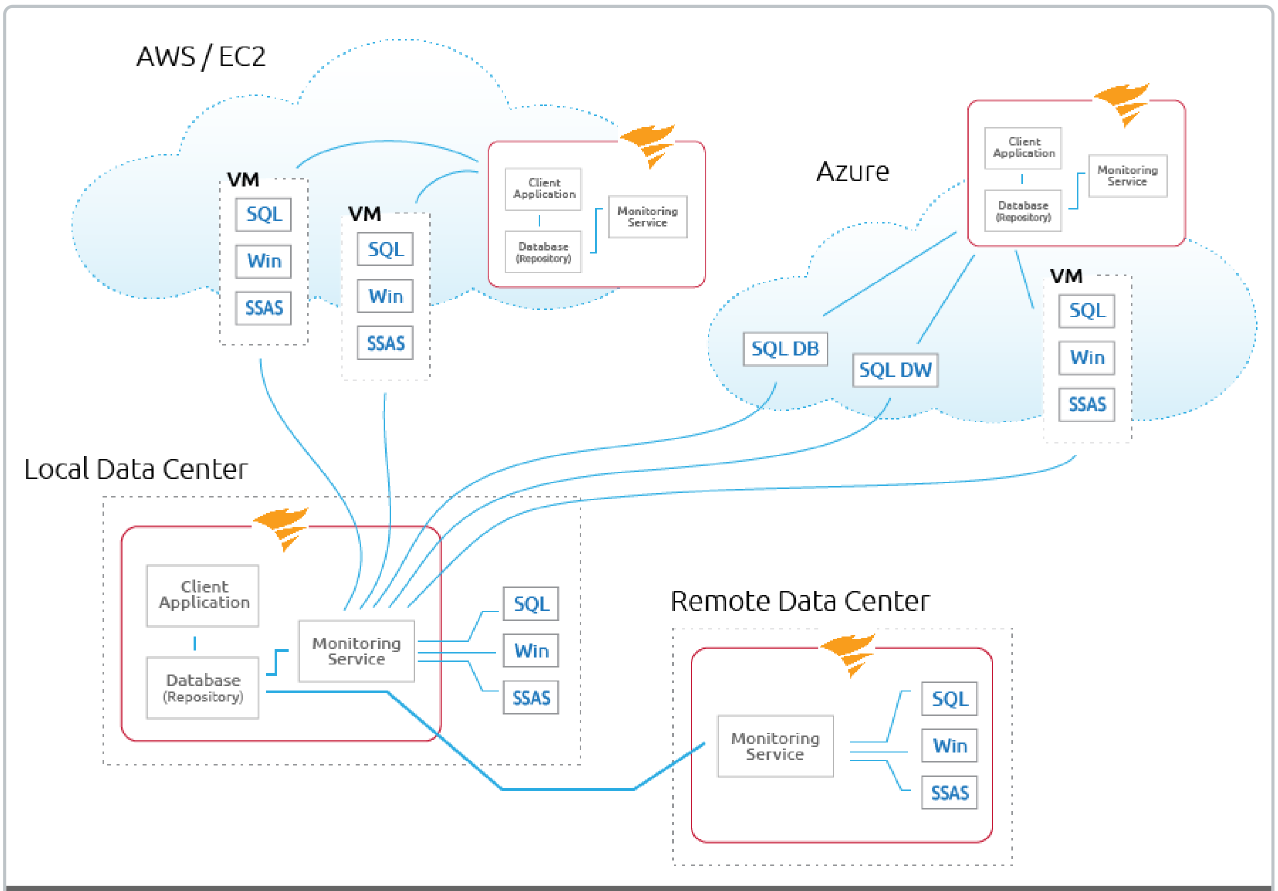
- SQL Sentry doesn't attempt to replace SQL Server Agent, Windows Task Scheduler or any other scheduling agents. SQL Sentry communicates with these schedulers to determine event status and collect history and performance information using a lightweight polling architecture.
- SQL Sentry doesn't require installed agents on each monitored target which dramatically reduces associated setup and maintenance overhead of agent-based systems. SQL Sentry also doesn't install a database on every monitored SQL Server. See the Watched Target Objects article for

details on which database objects are placed on a watched SQL Server target.

- There are times when the SQL Sentry client connects directly to the watched targets (monitored servers). For more information about when that happens, and how it can be secured, see the Client Security article.

SQL Sentry Architecture Examples

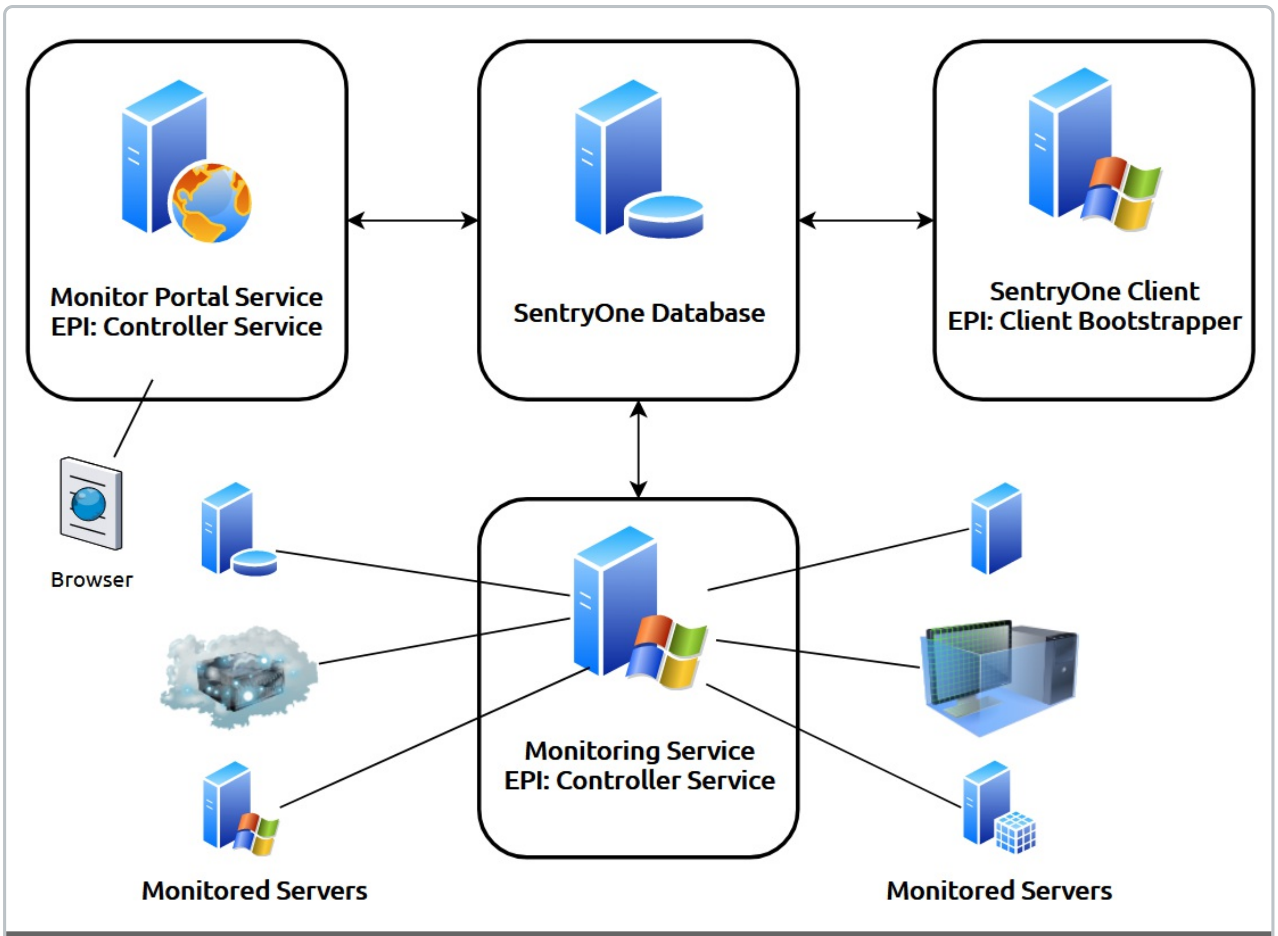
Overall Example



SQL Sentry Architecture Example

Example with SQL Sentry Portal and EPI

When using the EPI version of SQL Sentry, the components remain the same, but there are controller services and bootstrappers associated with some components. For example, when the SQL Sentry client is installed via EPI, it includes the client bootstrapper which allows for updates to be pushed out automatically to the clients during the upgrade process.



SQL Sentry architecture example with SQL Sentry installed across multiple machines (with EPI components when applicable)

Alerting and Response System

For its alerting and response system, SQL Sentry uses the concept of conditions and actions. Conditions describe the various states of monitored objects in your environment. Configure actions to take place when a condition is met.

All actions work on the principle of inheritance, meaning that an action configured in response to a condition being met at the global level (**All Targets** node in the **Navigator** pane), is automatically passed down to all applicable objects below it. This allows you to define global actions for the most common issues across your environment once, and have those passed down to every monitored target automatically.

Refine actions at each level as needed to determine what happens in response to events occurring in your server environment. Each instance type supports multiple conditions and actions.

Configuring actions globally significantly reduces the setup and configuration time required to implement notifications. For example, by enabling the **Send Email** action for the global **SQL Server Agent Job: Failure** condition, you automatically receive email alerts for any SQL Agent job failures across your enterprise. The only requirement is that the SQL Server instance and its jobs are watched by SQL Sentry. For a more detailed explanation of how conditions and actions work, see the Alerting and Response System topic.

Watching Instances and Objects

Throughout this document you'll also see the term **Watch** used frequently, in the context of watching instances or objects. When you have SQL Sentry **Watch** an instance or object through the context menu this simply means that SQL Sentry begins monitoring it.

Consider the following rules regarding watched instances and objects:

- When an instance is **Watched**, SQL Sentry monitors the instance and fires any applicable conditions for the instance based on its type.
- When an object is **Watched**, SQL Sentry monitors the object and fires conditions for the object based on its type.
- An instance can be **Watched** without watching any of its objects.
- If any object on an instance is set to **Watched**, the instance is automatically set to **Watched**.
- An object and its instance must be **Watched** to utilize SQL Sentry's queuing, chaining, and performance monitoring features.