MINION ENTERPRISE: INTRODUCTION

Introduction

Minion Enterprise is an enterprise solution for centralized and automated SQL Server management and alerting. This solution allows your database administrator to manage an enterprise of one, hundreds, or even thousands of SQL Servers from one central location. Minion Enterprise provides not just alerting and reporting, but centralized backups, maintenance, configuration, and enforcement.

Minion Enterprise was designed by Microsoft Certified Master database administrators (DBAs) specifically to address the issues that waste the most time and cause the most problems. Minion shrinks dozens of those one-at-a-time tasks down to a central process. And, it does this without the usual overhead of monitoring software.

This document is an introduction to Minion Enterprise and includes a list of top features, a quick start guide, configuration instructions, and a brief architecture overview.

Contents in Brief

Top 20 Features

Minion Enterprise (or just, ME) by MinionWare, LLC is an enterprise management solution for SQL Server. ME gives a database administrator an unprecedented amount of power over the enterprise, through extensive data collection, smart alerting, and the means to easily configure important settings across multiple instances. The DBA stands at the center of a wealth of information, with the levers to configuration close at hand.

Once installed and configured with a server list, ME begins to collect data about each server right away, and begins alerting as needed. Changes in settings, database lists, stopped and started services, and more are automatically pulled into the Collector tables.

Furthermore, Minion Enterprise allows you to configure settings on the local repository, and then it automatically pushes those changes out to the appropriate instances. Run a single UPDATE statement to set file growth rates for a single instance – or across the enterprise – or to enable the sp_configure option 'optimize for ad hoc workloads'.

Twenty of the very best features of Minion Enterprise are, in a brief:

1. **Fast, simple setup and configuration** – We have designed Minion Enterprise to set up as quickly as possible. Run the install, run one script, and insert server data to one table, and you're up and running: collecting data, alerting, and scripting out schemas automatically.
2. **Central Server List** – Keep a central list of SQL Server servers, and which applications they belong to. Beneficial for tracking your enterprise; auditing; and onboarding new staff.
3. **Central Configuration** – Central configuration and management for every aspect of Minion Enterprise. Make your changes inside Minion, and Minion then pushes your changes to all of the managed servers.
4. **Enforced Configuration** – Minion Enterprise provides you the option of enforcing server-level (sp_configure) configuration values. The policies are set centrally, so if someone then makes a change to the servers, Minion will change it back to conform to policy.
5. **Consolidated Alerts** – Centralized, consolidated alerts for all servers. This prevents "alert storming" support personnel, so alerts really mean something again.
6. **Low Footprint** – Minion Enterprise is a *management* system with central configuration, data collection, reporting, and alerting. It is not a traditional monitoring solution, and has very light resource overhead.
7. **Server Grouping** – Group servers by applications, and by service level (for example, Gold, Silver, and Bronze). This allows you to perform actions against only that application's servers, against only that application's production servers, against only Gold level servers, etc.
8. **Service Down Alerting** – Alert on SQL Server services that have stopped, across the whole enterprise.
9. **Database Properties and Backups** – Minion Enterprise tracks database properties, and alerts on missed backups.
10. **Track Table Sizes** – Query for data space used, index space used, and row counts for every table, in every database, in every managed instance.
11. **Search Error Logs** – Search SQL Server and SQL Agent error logs across the entire enterprise.

12. **Replication Tracking –** Track replication latency, and alert when it passes your custom threshold.
13. **Alert on Database Changes –** Query and alert on any database property change.
14. **Database Create/Drop History –** Retain history of database creation and deletion.
15. **Routine Database Object Scripting –** Script out all database objects regularly in order to retrieve schema changes. For example, if someone makes a change to a stored procedure on the wrong server, you have a record of the script before the change.
16. **Centralized SID Server –** Keep a central list of SQL Server logins, and their assigned ID number ("SID"). Standardizing login SIDs across your enterprise prevents the common "orphaned users" issue.
17. **Database User Reporting –** Centralized user account reporting.
18. **Windows Group Expansion** – See exactly who has what permissions via all Windows groups, even with nested groups.
19. **Disk Space Tracking and Alerting** – Report on the disk space usage across your enterprise, and define alert thresholds for specific disks.
20. **Integrates with free modules –** Minion Enterprise integrates very easily with the free modules, Minion Backup and Minion Reindex.

For more information on these, additional features and settings, and How To topics, see the sections **"How To" Topics**, and **Moving Parts**.

Quick Start

System Requirements and Installation

Minion Enterprise has a few system requirements (note that these do not all apply to the *managed* servers). We do recommend that you install on a dedicated instance of SQL Server: a modest virtual machine is the ideal scenario.

System requirements:

- SQL Server 2012 or above. (Managed instances can be SQL Server 2005 or above.)
- The SQL Agent service account requires sysadmin rights on all managed SQL Server instances, and access to run system-level WMI queries on all managed servers (which can be accomplished by granting the SQL Agent service account local administrator rights on each managed server). **Note**: Check your company's security policies for compliance.
- The sp_configure setting **xp_cmdshell** must be enabled*.
- PowerShell 2.0 or above; execution policy set to **RemoteSigned**.

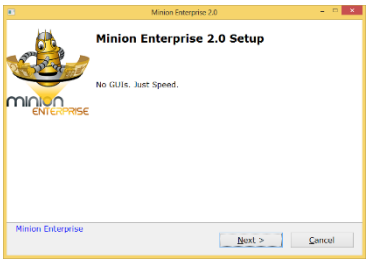
** xp_cmdshell can be turned on and off with the database PreCode / PostCode options, to help comply with security policies.*

System recommendations:

- Install on a dedicated server, or dedicated virtual machine
- 6-8GB memory

- 4 CPU
- 200-300 GB of disk space
- Windows 2012 R2

To install Minion Enterprise:



1. Run MinionEnterprise.exe:
    a. Review and agree to the license.
    b. Click Finish.
2. Obtain a license:
    a. To purchase a license, email us at Support@MidnightDBA.com with the number of servers you'd like to manage, and the name of your Minion server.
    b. Or, to obtain a 90 day trial license, email Support@MidnightDBA.com with the name of your Minion server.
3. Install the license:
    a. Save the License.dll file to **C:\MinionByMidnightDBA\Collector**
    b. Run the following from a command prompt:
    C:\MinionByMidnightDBA\Collector\License.exe Install
4. Add your email to the **dbo.EmailNotification** table:
```
INSERT INTO dbo.EmailNotification ( EmailAddress, Comment )
SELECT 'Support@MidnightDBA.com' , 'Administrator';
```
5. Edit and run the mail setup script:
    a. Open the mail setup script for editing: **C:\MinionByMidnightDBA\MinionMail.sql**
    b. Set the @MailServer variable to your SMTP address.
    c. Optionally, you can change the various **@display_name** values.
    Note: You do not have to change the @Email… and @ReplyTo… variables, as you don't reply to alert emails.
    d. Run the edited MinionMail.sql script.

Initial Configuration

The initial configuration is remarkably simple: Just enter servers to be managed in the **dbo.Servers** table.

To set up managed servers in Minion Enterprise, enter server information into the dbo.Servers table in any way you choose: INSERT statements, an Excel file import, with an Active Directory lookup, or any other method of pulling data into a table.

Each server only requires you to set three pieces of information and six flags in the dbo.Servers table to begin. The information you must configure is:

- **ServerName** – The name of the SQL Server instance; or, if it is a cluster, the name of the SQL Server virtual instance. Note that Minion Enterprise can monitor Windows servers that do not have an instance of SQL Server installed.

- **ServiceLevel** – All servers must be assigned a service level. We recommend using Gold, Silver, and Bronze to group your servers into levels; however, you may choose system method you like.
- **Port** – If your SQL Server instance is not on the standard port (1433), you must enter the port number.

The flags to configure are:

- **IsSQL** – Whether this is a SQL Server server or not. Minion Enterprise can perform certain kinds of monitoring (for example, disk space) on non-SQL Server servers.
- **BackupManaged** – Enables backup statistics collection and alerts.
- **DiskManaged** – Enables disk statistics collection and alerts.
- **IsServiceManaged** – Enables service data collection and alerts.
- **SPConfigManaged** – Enables sp_configure data collection and alerts.
- **IsActive** – Whether this server should be actively managed or not. For example, you may set isActive to 0 to stop managing a single instance that is no longer in use. Minion Enterprise still retains the server information in dbo.Servers, but no longer collects or alerts for that server.

So for example, we could configure 'YourServer' like this:

```
INSERT  INTO dbo.Servers
        ( ServerName ,
          Port ,
          ServiceLevel ,
          IsSQL ,
          BackupManaged ,
          DiskManaged ,
          IsServiceManaged ,
          SPConfigManaged ,
          IsActive
        )
SELECT  'YourServer' AS ServerName ,
        1433 AS Port ,
        'Gold' AS ServiceLevel ,
        1 AS IsSQL ,
        1 AS BackupManaged ,
        1 AS DiskManaged ,
        1 AS IsServiceManaged ,
        1 AS SPConfigManaged ,
        1 AS IsActive ;
```

Once this data is entered, Minion Enterprise will automatically:

- **Collect additional information about each server, from the managed servers themselves.** For more information, see the documentation for jobs CollectorServerInfoGetGOLD, CollectorServerInfoGetSILVER, and CollectorServerInfoGetBRONZE.
- **Begin collections on each of the listed servers.** See the "Quick Tour of the Data Collections" section below for more information.

- **Begin alerting for each of the listed servers.** For more information, see the documentation for "Alert" jobs.
- **Begin scripting out all schemas for "Gold" level servers on a daily basis.** This is a massively useful feature, as a safeguard against schema level mistakes (such as an incorrectly modified view, or a mistakenly dropped index, or almost any schema object that is modified or dropped). By default, schema are scripted out to C:\MinionByMidnightDBA\DBScriptBackups\. For more information, see the section titled "Change the Default Location for Scripted Objects".

About Data Collections

Minion Enterprise gathers data from all managed servers into tables in the **Collector** schema. The moment you configure an instance in the dbo.Servers table, Minion Enterprise begins pulling in a stunning array of data: service status, disk space, backups, database properties, logins and users, table sizes, and on and on. All of this data from across your enterprise is now updated and available in one central repository, ready for alerts, and reporting, and planning for the future.

Minion Enterprise further simplifies matters by providing a view for many of the Collector tables; these views provide only the most recent collection for their respective tables.

Here in the Quick Start, we review just a handful of these data collections.

Server Environment Data	The Collector.ServersOSDetail table holds information about each server's memory, OS version, install date, and more. Use the view **Collector.ServersOSDetailCurrent** for the latest data. To view the latest disk space usage data, query the view **Collector.DriveSpaceCurrent**. It provides the latest InstanceID, server name, drive name, drive capacity, drive free space, service level, and more.
SQL Server Environment Data	To check the SQL Server services' latest Running/Stopped status for all servers, query the view **Collector.ServiceStatusCurrent**. It provides the latest InstanceID, server name, service level, service name, service status, and more. Similarly, check on current service accounts using **Collector.ServiceAcctCurrent**; retrieve up-to-date sp_configure settings via **Collector.InstanceConfigCurrent**; and recent error log collections in **Collector.ErrorLogCurrent**.
Database Information	Get a current list of databases from **Collector.DatabasesCurrent**. And for each database, the Collector.DBProperties table holds a significant amount of data, including the database owner; last full, log, or diff backup; which databases have auto-shrink enabled; database collations; case sensitivity; database size on disk; and more. To see the most recent collection of database properties, query **Collector.DBPropertiesCurrent**. For current information on database file properties and growth rates, see **Collector.DBFilePropertiesCurrent** and **Collector.FileGrowthRateCurrent**. The view Collector.LogspaceCurrent shows the latest log space usage. The Collector.TableProperties table provides data space used, index space used, and rowcounts for every table, in every database, in every managed instance. For

	the most recent collection of table size data, query **Collector.TablePropertiesCurrent**.
Security and Encryption	You can see an up to date list of all logins across all servers using the view **Collector.LoginsCurrent**, and the list of all databases users across all servers using **Collector.DBUsersCurrent**. The Collector.ADGroupMember table expands AD groups into their constituent members (AD logins and sub-groups). In an upcoming service pack, we will provide a useful view that better ties this information to the rest of the system.

Retrieving Server Information

Minion Enterprise provides stored procedures and views to help you find pertinent information. Two of these stored procedures are Servers.ByName and Servers.ByID.

Search using Servers.ByName

The Servers.ByName procedure allows you to search for servers by name. To search for a specific server name – for example, "YourServer" – execute the procedure with the @ServerName parameter:

```
EXEC [Servers].[ByName] 'YourServer';
```

To search for a partial name – for example, any server with "Your" in the name – execute the procedure with the @ServerName and @Fuzzy parameters:

```
EXEC [Servers].[ByName] 'Your', 1;
```

In either case, the procedure will return a great amount of data for YourServer, including application information (if any), InstanceID, IP and port, service level, and other collected and configured data.

Search using Servers.ByID

The Servers.ByID procedure allows you to search for servers by instance ID. To search for a server by ID, execute the procedure with the @InstanceID parameter:

```
EXEC [Servers].[ByID] @InstanceID = 2;
```

The procedure will return a great amount of data for the server with InstanceID=2, including application information (if any), InstanceID, IP and port, service level, and other collected and configured data.

Tour of the ME Modules

Minion Enterprise gathers data from all managed servers into tables in the **Collector** schema. The moment you configure an instance in the dbo.Servers table, Minion Enterprise begins pulling in a stunning array of data: service status, disk space, backups, database properties, logins and users, table sizes, and on and on. All of this data from across your enterprise is now updated and available in one central repository, ready for alerts, and reporting, and planning for the future.

Minion Enterprise further simplifies matters by providing a view for many of the Collector tables; these views provide only the most recent collection for their respective tables.

Here in the Quick Start, we review just a handful of these data collections.

Server Environment Data	**OS Details:** The OS Detail module collects information about each server's memory, OS version, install date, and more. Use the view **Collector.ServersOSDetailCurrent** for the latest data. **Drive Space:** To view the latest disk space usage data, query the view **Collector.DriveSpaceCurrent**. It provides the latest InstanceID, server name, drive name, drive capacity, drive free space, service level, and more. **Startup Accounts:** The ME Startup Account module collects SQL Server startup account information. This is an excellent way to see the impact of an account password change. See the latest collection of service account data using the **Collector.ServicePropertiesCurrent** view.
SQL Server Environment Data	**Service Status:** To check the SQL Server services' latest Running/Stopped status for all servers, query the view **Collector.ServiceStatusCurrent**. It provides the latest InstanceID, server name, service level, service name, service status, and more. **Instance Configuration:** Retrieve up-to-date sp_configure settings via **Collector.InstanceConfigCurrent**. **Error Log Searches:** You can set up specific error log search terms in the dbo.ErrorLogSearch table. The ME Error Log module gathers any matches and logs them in the Collector.ErrorLog table. See the **Collector.ErrorLogCurrent** view for the latest results. **Wait Statistics:** The **Collector.WaitStatsCurrent** view shows the latest collection of wait stats data, including the percentage of the WaitType for each collection period.
Database Information	**Databases:** The Database module collects a list of databases per server. Get the current list of databases from **Collector.DatabasesCurrent**. **DB Properties:** For each database, the Database Properties module collects a significant amount of data, including the database owner; last full, log, or diff backup; which databases have auto-shrink enabled; database collations; case sensitivity; database size on disk; and more. To see the most recent collection of database properties, query **Collector.DBPropertiesCurrent**. **Database Files:** For current information on database file properties and growth rates, see **Collector.DBFilePropertiesCurrent** and **Collector.FileGrowthRateCurrent**. The view **Collector.LogspaceCurrent** shows the latest log space usage. **Script Schemas**: Minion Enterprise automatically scripts out all objects for "Gold" level servers, on a daily basis. You can use this as a safeguard against schema level mistakes; a basic, secondary form of version control; and more. Objects are scripted out to files on the hard drive, so there is no associated view with this module.

	Objects: The ME Objects module collects sys.objects data from each database. This is extremely useful for determining when objects come and go from databases, and to investigate when an object might be missing from certain servers. Get the most recent collection from the **Collector.SysObjectsCurrent** view. **Tables:** The Table Properties module gathers data space used, index space used, rowcounts, and much more for every table, in every database, in every managed instance. For the most recent collection of table data, query **Collector.TablePropertiesCurrent**. **Columns:** The **Collector.TableColumnsCurrent** view shows the latest collection of table column data. You use this data to, for example, detect changes to tables over time, compare the data types for similarly named columns across your enterprise, or search for common issues (such as VARCHAR(1), or deprecated data types). **Indexes:** Get detailed index information from the Index Stats module. It collects all data from sys.indexes, plus the list of indexed columns (and included columns), and more. This allows you to perform detailed, enterprise-wide index analysis. Check the **Collector.IndexStatsCurrent** view for the latest set of index information.
Maintenance and Backups	**Backups:** Minion Enterprise can log the latest times for backups performed by Minion Backup. For the most recent collection of database properties – including latest backup times - query **Collector.BackupLogDetailsCurrent**. **Index Maintenance:** You have the option of integrating Minion Reindex data into Minion Enterprise. Once Minion Reindex is integrated, you can find the most recent collection of index maintenance logs in the **Collector.IndexMaintLogCurrent** view. **CheckDB:** Once Minion CheckDB is released (planned for late 2015), you will have the option to integrate it into Minion Enterprise.
Security and Encryption	**Logins:** See an up to date list of all logins across all servers using the view **Collector.LoginsCurrent**. Use the Login Password Strength feature to test the password strength of all logins in your enterprise, using the Password Audit feature. You can also use the SID Server feature to standardize login SIDs across your enterprise, and prevent the common orphaned users issue. **Server Role Members:** The **Collector.ServerRoleMemberCurrent** view displays the latest collection of logins, by server role. **AD Group Members:** The **Collector.ADGroupMemberCurrent** view expands the latest collection of AD groups into their constituent members (AD logins and sub-groups). **Database Principals:** Minion Enterprise also collects database security principals. Get the latest collection by querying the view **Collector.DatabasePrincipalsCurrent**. **Database Users:** The **Collector.DBUsersCurrent** view provides the latest collection of database users. Use it, together with the **Collector.DBUsersPrevious** view, to discover when users are added and deleted from specific databases.

Replication Latency	**Replication:** Minion Enterprise gathers replication latency data for all subscriptions. The **Collector.ReplLatencyCurrent** view displays the latest collection. Note that this is one of the few modules that requires some setup.
Minion Admin	**Data Archiving:** The Data Archiving module cleans out old, collected data automatically. Retention periods are configurable.

Full Configuration

This section assumes you have already completed the installation and initial configuration outlined in the Quick Start guide above, under "Initial Configuration". This initial configuration gets Minion Enterprise up and running. There are a handful of additional, optional configuration items you can use to fine-tune the ME operation, including:

- Configure existing Minion Reindex installations in the **dbo.ServerMgmtDB** table.
- Configure the scripts path in the "ServerConn.ps1" configuration file.
- Push configuration changes to managed instances
- Enforce configuration settings for managed instances
- Configure alerting

Integrate Minion Reindex

Minion Enterprise can integrate your existing Minion Reindex installations on managed servers, pulling in maintenance information to the central repository.

We will go over the basic steps used to integrate Minion Reindex with Minion Enterprise, and then we'll walk through an example. The basic steps to integrate an existing instance of Minion Reindex with ME are:

1. Ensure that the managed instance is in the dbo.Servers table in ME.
2. Add a row for each server to dbo.ServerMgmtDB. This is only necessary if Minion Reindex is not installed in the master database.
3. Import reindex settings to ME using the MRSettingsImport.exe script.
4. Verify the import.
5. Configure the Index MinionTriggerPath for each managed server, and set them to be pushed out to the MR instances.

Once this is complete, Minion Enterprise will push the trigger path settings to each managed instance; each successive run of Minion Reindex will generate a trigger file (per database) to that location; and the MRHistoryProcess job will use the trigger file to pull new index maintenance history to the ME repository. The data that is pulled into Minion Enterprise is stored in two tables:

- Collector.IndexMaintLog
- Collector.IndexMaintLogDetails

Example

Let's now walk through an example of this process. Minion Enterprise is installed on a server named "*MinionProd*". We have a managed server named "*ManagedServer*" which already has Minion Reindex (or as we call it, MR) installed and running. To integrate the *ManagedServer* Minion Reindex with ME:

One: Instance in dbo.Servers. Check that *ManagedServer* has a row in the dbo.Servers table on *MinionProd* (in the Minion database). If *ManagedServer* is not in the table, insert it to dbo.Servers as described in "Initial Configuration" above.

Two: Management databases in dbo.ServerMgmtDB. Check to see which database Minion Reindex is installed in, on *ManagedServer*. If it were in master, no action wold be needed here. However, we see that MR was installed in the "DBA" database, so we must insert a row for *ManagedServer* to dbo.ServerMgmtDB. For our example, we would run the following:

```
INSERT  INTO ServerMgmtDB ( InstanceID , MgmtDB )
SELECT  InstanceID ,
        'DBA' AS MgmtDB
FROM    dbo.Servers
WHERE   ServerName = 'ManagedServer';
```

Three: Import Minion Reindex settings. We have configured *ManagedServer* with a Gold service level. To import reindex settings to ME, open a command prompt on *MinionProd*, and run the import script, passing an input parameter of "Gold":

C:\MinionByMidnightDBA\Collector\MRSettingsImport.exe Gold

Four: Verify import. Verify that *ManagedServer*'s "MinionDefault" settings were imported to **dbo.IndexSettingsDBDefault**. If any MR settings were defined at the database level, they also will have been imported, to the dbo.IndexSettingsDB table.

Five: Configure the trigger path. Choose or create a network share to hold trigger files; in our example, we will use "\\MinionProd\TriggerPath\". We must update the trigger path for the instance, in both the default row (in dbo.IndexSettingsDBDefault), *and* in any database-specific rows (in dbo.IndexSettingsDB):

```
-- Update the ManagedServer default row:
UPDATE  dbo.IndexSettingsDBDefault
SET     MinionTriggerPath = '\\MinionProd\TriggerPath\' ,
        Push = 1
WHERE   InstanceID = ( SELECT   InstanceID
                       FROM     dbo.Servers
                       WHERE    ServerName = 'ManagedServer'
                                                );

-- Update the ManagedServer DB-specific rows:
UPDATE  dbo.IndexSettingsDB
SET     MinionTriggerPath = '\\MinionProd\TriggerPath\' ,
        Push = 1
```

```
WHERE   InstanceID = ( SELECT   InstanceID
                       FROM     dbo.Servers
                       WHERE    ServerName = 'ManagedServer'
                     );
```

Because we set the Push field to 1, the changes will be pushed out to the MR instance.

Note: If you have more than one Minion module – for example, Minion Reindex and Minion Backup (MR and MB) – Minion Enterprise will only integrate them all *if they are installed in the same database* on the managed server. For example, if Server2 has MR and MB installed in master, then Minion Enterprise can integrate both modules on that managed server, once you've follwed the process above.

Change the Default Location for Scripted Objects

To move the default location of scripted objects, configure the "$DBScriptBasePath" variable in the **ServerConn.ps1** configuration file, located at **C:\MinionByMidnightDBA\Includes**. For example, you might change the default code:

```
$DBScriptBasePath = "C:\MinionByMidnightDBA\DBScriptBackups"
```

To use a different drive:

```
$DBScriptBasePath = "D:\MinionByMidnightDBA\DBScriptBackups"
```

IMPORTANT: Be sure that your target directory exists.

To disable this feature entirely, disable the "CollectorDBScriptGOLD" job.

Push Configuration Changes to Managed Instances

You can set and push certain types of settings to managed instances, by setting the "Push" field to 1. After you make these changes, ME will pick up this change and push it to the appropriate instances automatically:

- **dbo.DBFilePropertiesConfig** – You can set database file properties for files on one or more servers using this table; just update the appropriate settings and set Push=1.
- **dbo.InstanceConfigValue** – You can set sp_configure values for one or more servers using this table; just update the appropriate settings and set Push=1.
- **dbo.IndexSettingsDBDefault** – This table holds the default level Minion Reindex settings for all the databases on an instance. Make changes for one or more managed servers that have an installed Minion Reindex module, using this table.
- **dbo.IndexSettingsDB** – This table holds the database level Minion Reindex override settings for specific databases on an instance. Make changes for one or more managed servers that have an installed Minion Reindex module, using this table.
- **dbo.IndexSettingsTable** – For Minion Reindex instances, holds the table level Minion Reindex override settings for specific tables on an instance. Make changes for one or more managed servers that have an installed Minion Reindex module, using this table.

Enforce Configuration Settings for Managed Instances

Minion Enterprise allows you to not only push, but enforce the sp_configure values for a SQL Server instance. Let's take a quick example to illustrate how this works;

1. You have configured 'optimize for ad hoc workloads' to 1 for all managed instances, in the dbo.InstanceConfigValue table.
2. For that particular setting, you have set Push=1 and Action='Enforce' in the dbo.InstanceConfigValue table.
3. Minion Enterprise automatically pushes that configuration to all managed instances.
4. The following week, someone executes sp_configure to set 'optimize or ad hoc workloads' to 0 on three of the managed servers.
5. Minion Enterprise will automatically change the configuration back for those instances.

Configure Alerting

Minion Enterprise comes installed with jobs and pre-configured alert thresholds for important events, such as missed backups and dropped databases.

To configure thresholds for drive space alerts, use the following tables:

- **dbo.DriveSpaceThresholdServer** – This is an alert configuration table. It holds default disk space thresholds for an entire server – in other words, thresholds for all the drives on a server.
- **dbo.DriveSpaceThresholdDrive** – This is an alert configuration table. It holds default disk space thresholds for individual drives. Each drive can have its own threshold. (Note that drive level settings take precedence over server level settings; so if YourServer has an entry in dbo.DriveSpaceThresholdServer, an entry for drive YourServer's drive D: in dbo.DriveSpaceThresholdDrive will override that setting for that drive.)

To configure thresholds for backups space alerts, use table **dbo.DBMaint**.

- The "BackUpAlertThresholdHrs" column allows you to set the backup alert threshold in hours for both full and differential backups.
- The "LogBackupAlertThresholdMins" column allows you to set the backup alert threshold in minutes for log backups.

The following alert items have no threshold configuration:

- New and dropped databases.

To configure thresholds for replication latency, set the following values in the **dbo.ReplPublisher** table:

- InstanceID – the instance ID of the instance in question
- DBName – the database name in question
- PublName – the publication name.
- SendAlert – enalble alerting

- AlertMethod – method of alerting (i.e., 'Secs')
- AlertValue – the alert threshold, in number of seconds

Architecture Overview

Minion Enterprise is made up of SQL Server stored procedures, tables, jobs, executables, and configuration ("config") files:

- Tables store configuration and log information.
- Stored procedures and executables perform various operations.
- Config files provide information to executables.
- Jobs execute those operations on a schedule.

Note: The Minion Enterprise installer creates a Minion database, and installs ME in that new database.

Future editions of this document will contain more information about architecture, including configuration settings hierarchy, logging, and data archiving.

Data Archiving

The tables in Minion hold collection data from all the servers in your network. This data ages out and becomes too large to handle reasonably, so it's necessary for Minion to clean up after itself. Therefore, Minion Enterprise includes archival process for the collector data, and any other data that needs archiving.

About Us

Minion Enterprise is a creation of Jen and Sean McCown, owners of MinionWare, LLC and MidnightSQL Consulting, LLC.

In our "**MidnightSQL"** consulting work, we perform a full range of databases services that revolve around SQL Server. We've got over 30 years of experience between us and we've seen and done almost everything there is to do. We have two decades of experience managing large enterprises, and we bring that straight to you. Take a look at www.MidnightSQL.com for more information on what we can do for you and your databases.

Under the "**MidnightDBA"** banner, we make free technology tutorials, blogs, and a live weekly webshow (DBAs@Midnight). We cover various aspects of SQL Server and PowerShell, technology news, and whatever else strikes our fancy. You'll also find recordings of our classes – we speak at user groups and conferences internationally – and of our webshow. Check all of that out at www.MidnightDBA.com

We are both "MidnightDBA" and "MidnightSQL"…the terms are nearly interchangeable, but we tend to keep all of our free stuff under the MidnightDBA banner, and paid services under MidnightSQL Consulting, LLC. Feel free to call us the MidnightDBAs, those MidnightSQL guys, or just "Sean" and "Jen". We're all good.

minion
ENTERPRISE

Contents

For more information, see MidnightSQL.com/Minion or email us at Support@MidnightDBA.com today!