

1. Database basics

MonitorMagic supports recording monitoring data to a database for graphing and reporting purposes. This document describes some best practice examples in using a database with MonitorMagic.

MonitorMagic currently supports 3 databases:

- **Microsoft SQL Server®**
- **Microsoft Desktop Engine® (MSDE)**
- **Microsoft Access®**

Tools4ever strongly recommends using either SQL Server or the MSDE for the best results, since Microsoft Access does not provide the same level of stability, scalability and performance. The advantage of MSDE is that it is available as a free download on the Microsoft website. If you plan to use SQL Server, you may have to purchase an extra license if you plan on having a dedicated database for MonitorMagic.

Note 1: the pre-configured reports on MonitorMagic are exclusively for use with SQL Server and MSDE. They will not work when using Microsoft Access. MonitorMagic does not provide functionality for database backup or optimization.

Note 2: MonitorMagic only supports a US English version of SQL Server or MSDE. Running any other language version may result in unexpected behavior.

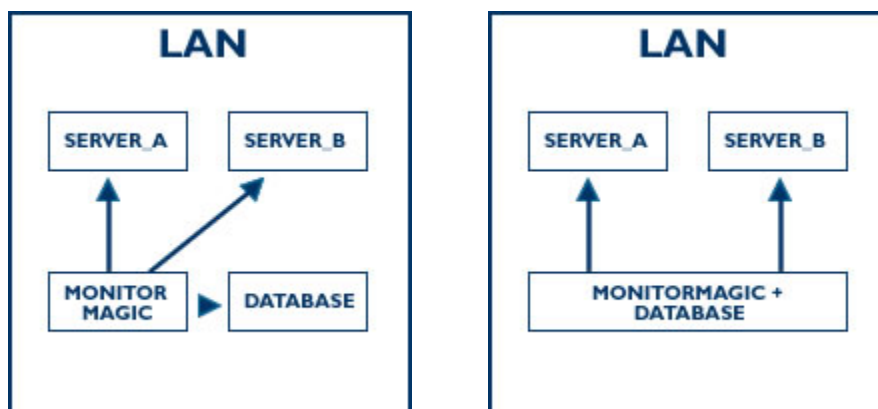
Note 3: MonitorMagic supports both Windows and SQL authentication when creating a new database. When choosing the SQL authentication option using username and password, make sure the SQL Server is set to allow mixed authentication, otherwise this option will result in errors.

2. Central or distributed database

The choice of having a central or distributed database model is a fairly easy one; always opt for the central model per LAN. The ideal scenario is that each MonitorMagic service is monitoring approximately 50 servers and archiving data for about 25 servers, and that each service is connected to a central SQL Server database.

The central database model simplifies configuration and implementation, as there is a single source for using graphs, reports and report distribution. When monitoring multiple LANs, use a distributed model of central databases. This means that you have a central database per LAN and use a single source for distributing reports per LAN.

Note: more information on placing databases in local/remote environments can be found in the *MonitorMagic Connection Reference*.



3. Using SQL authentication when service is running in DMZ

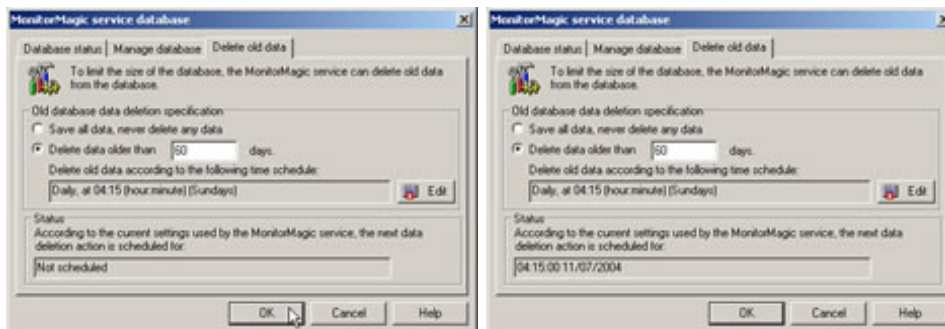
- Configure the SQL Server to support both Windows and SQL authentication, be sure to restart the SQLServer services afterwards. Note that modifying this setting can take a while; please be patient for the operation to complete.
- Modify the data source, switch from Windows authentication to SQL authentication, and make sure that the MonitorMagic database is the default database for the data source.
- In MonitorMagic, go to the service configuration, and then the advanced -> database configuration. Use the Configure button and select the existing data source which was preset to use Windows authentication. Make sure the data source is selected and the radio button is enabled. In the lower portion of the screen, make sure that you include the username and password for the SQL account:
 DSN=MonitorMagicSql;UID=<sqlaccountname>;PWD=*****
- The existing database will now be used and the status will be shown as 'Ok' in the MonitorMagic Control Center.

4. Automatic database cleanup

MonitorMagic is able to perform automatic database maintenance at scheduled times. The maintenance operation can optionally remove all data older than a number of days. This option is disabled by default, Tools4ever recommends you enable this option and set it to the lowest possible value, for instance 30. When using reporting, you already have archived 4 full reports before you start to remove old data, which Tools4ever regards as a safe margin.

To enable database cleanup, access the service configuration by right-clicking on the service on the MonitorMagic control center and select "**Configure service**". Move to the "**Advanced**" tab, then click "**Configure database**". The settings you need to configure are located in the "**Delete old data**" tab. When you configure and confirm your settings, then afterwards return to the same tab, you will notice that the final scheduling date for the cleanup has been set, similar to the screenshots below.

Note: at the time of writing, MonitorMagic does not automatically clean up any data related to event archiving. We plan to remedy this in our next maintenance release. When using MSDE, make note that the total size of the instance is limited to 2GB. Tools4ever strongly recommends monitoring your database growth, more information on this subject can be found later on in this document.



5. Limit database growth & manual database cleanup

To effectively control the database growth, especially when using event archiving, Tools4ever recommends several adjustments to the MonitorMagic database in SQL Server, which involves limiting the size of the transaction log.

Limit transaction log size:

1. Open the SQL Server Enterprise Manager
2. Access the properties for the MonitorMagic database
3. In the tab "Transaction log", enter a value for "Restrict growth (MB):".
4. In the tab "Options", set the Recovery method to "Simple".
5. Open the Query Analyzer, connect to the MonitorMagic database
6. In an open query window, type "DBCC SHRINKDATABASE (<dbname>)"

Stored procedures to perform manual database cleanup:

To run, download the file below and use it to create a new stored procedure in SQL Server. Then, using the Query Analyzer, run the stored procedures as outlined below.

BE EXTREMELY CAREFUL when running this script. Tools4ever is not responsible for any loss of data whatsoever. The code snippets below are stored procedures, which have to be imported into SQL Server or MSDE before they can be used.

1: DeleteDataOlderThanNumberOfDays:

SYNTAX: EXEC DeleteDataOlderThanNumberOfDays <numberofdays>
<numberofdays> = number of days for which you want to keep data. All data older than the current date - <numberofdays> is deleted.

```
CREATE PROCEDURE DeleteDataOlderThanNumberOfDays
    @Days int
AS
DECLARE @DeleteDate datetime

SET @DeleteDate = DATEADD(d, -@Days, GetDate())

DELETE FROM MmLog_EventRecord WHERE LogTimeGenerated < @DeleteDate
DELETE FROM MmLog_SourceName WHERE SourceID NOT IN (SELECT LogSourceID FROM
MmLog_EventRecord)
DELETE FROM MmLog_UserName WHERE UserID NOT IN (SELECT LogUserID FROM
MmLog_EventRecord)
DELETE FROM MmLog_Description WHERE DescriptionID NOT IN (SELECT LogDescriptionID FROM
MmLog_EventRecord)

DELETE FROM MonitorDiskData WHERE DataDate < @DeleteDate
DELETE FROM MonitorDiskDefinition WHERE DbDiskID NOT IN (SELECT DbDiskID FROM
MonitorDiskData)

DELETE FROM MonitorPerformance_CounterData WHERE DataDate < @DeleteDate
DELETE FROM MonitorPerformance_CounterDefinition WHERE DbPerformance_counterID NOT IN
(SELECT DbPerformance_counterID FROM MonitorPerformance_CounterData)

DELETE FROM MonitorPingData WHERE DataDate < @DeleteDate
DELETE FROM MonitorPingDefinition WHERE DbPingID NOT IN (SELECT DbPingID FROM
MonitorPingData)

DELETE FROM MmGeneral_ComputerName WHERE ComputerID NOT IN (SELECT LogComputerID FROM
MmLog_EventRecord)
GO
```

2: DeleteDataForSpecifiedComputer:

SYNTAX: EXEC DeleteDataForSpecifiedComputer <computer>

This will delete all monitoring and event archiving data associated with a computer.

```
CREATE PROCEDURE DeleteDataForSpecifiedComputer
    @ComputerName nvarchar(64)
AS
DECLARE @ComputerID int
DECLARE @LogDescriptionID int
DECLARE @LogSourceID int
DECLARE @LogUserID int

SET @ComputerID = (SELECT ComputerID FROM MmGeneral_ComputerName WHERE ComputerName =
@ComputerName)
DELETE FROM MmLog_EventRecord WHERE LogComputerID = @ComputerID
DELETE FROM MmLog_SourceName WHERE SourceID NOT IN (SELECT LogSourceID FROM
MmLog_EventRecord)
DELETE FROM MmLog_UserName WHERE UserID NOT IN (SELECT LogUserID FROM
MmLog_EventRecord)
DELETE FROM MmLog_Description WHERE DescriptionID NOT IN (SELECT LogDescriptionID FROM
MmLog_EventRecord)

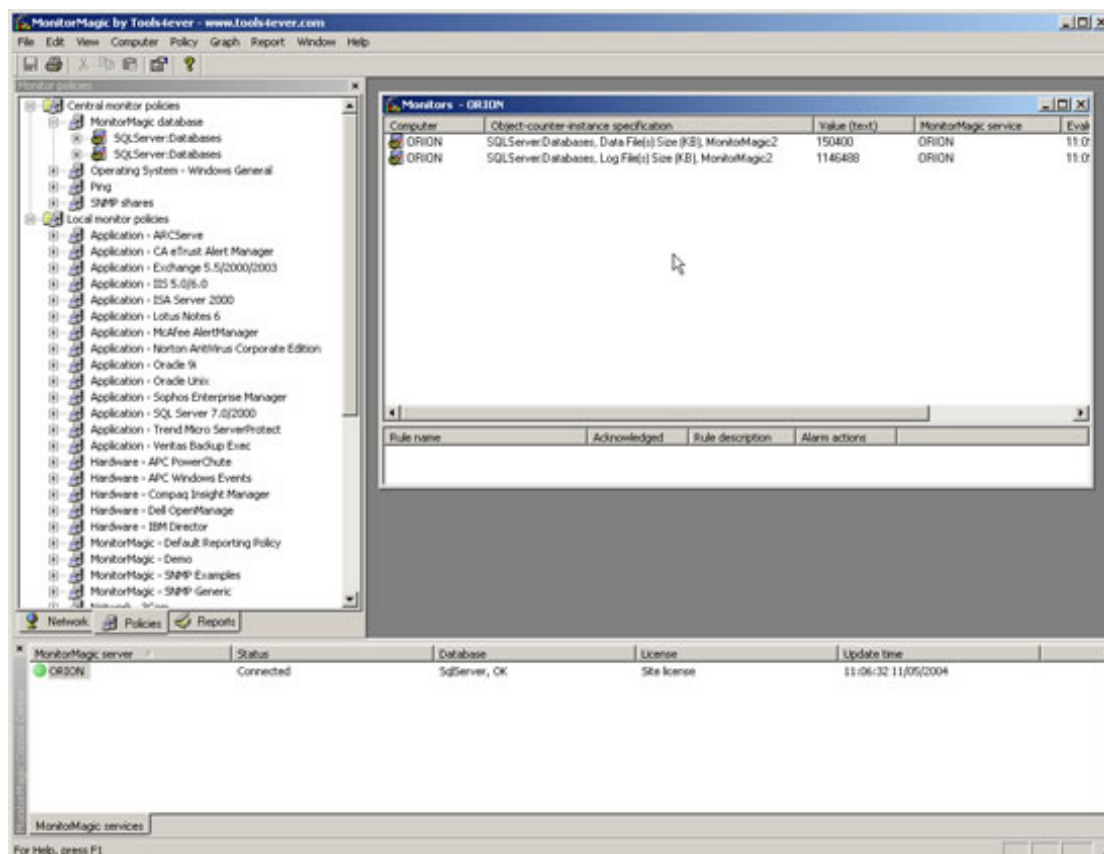
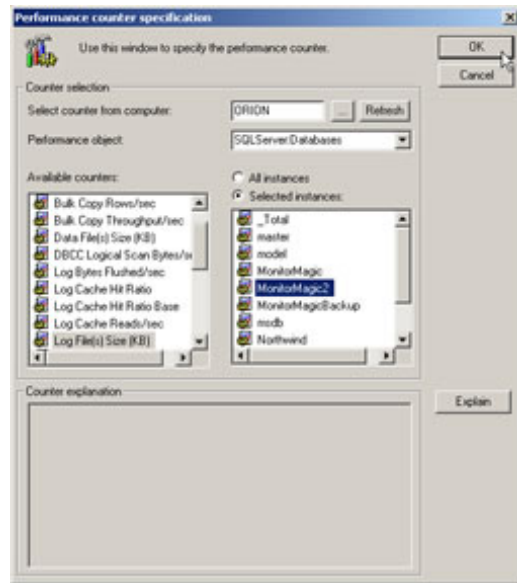
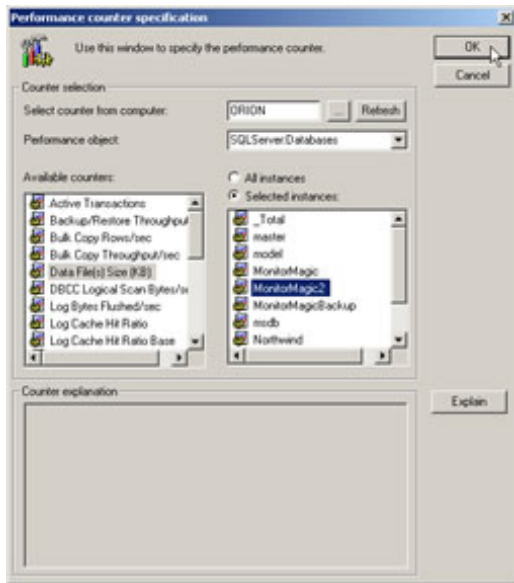
DELETE FROM MonitorDiskData WHERE DbDiskID IN (SELECT DbDiskID FROM
MonitorDiskDefinition WHERE DbComputer = @ComputerName)
DELETE FROM MonitorDiskDefinition WHERE DbComputer = @ComputerName

DELETE FROM MonitorPerformance_CounterData WHERE DbPerformance_counterID IN (SELECT
DbPerformance_counterID FROM MonitorPerformance_CounterDefinition WHERE DbComputer =
@ComputerName)
DELETE FROM MonitorPerformance_CounterDefinition WHERE DbComputer = @ComputerName

DELETE FROM MonitorPingData WHERE DbPingID IN (SELECT DbPingID FROM
MonitorPingDefinition WHERE DbHost = @ComputerName)
DELETE FROM MonitorPingDefinition WHERE DbHost = @ComputerName
GO
```

6. Monitoring database growth

When running both SQL Server and MSDE, you may want to keep a close watch on the growth of the MonitorMagic database. This can easily be accomplished by creating a new policy with 2 performance counter monitors. Configure the performance counter monitors to monitor the counters as shown in the screenshots below (the name of the instance can differ for your environment, always choose the name MonitorMagic, and if applicable, followed by the highest number). After you apply the policy, you will see the values in KB for both the data and log file of the MonitorMagic database.



7. Troubleshooting database contents

Use the below SQL query to analyze the start and end dates for which resources have been monitored or archived. The output will contain in detail which computers have been included in monitoring and event archiving jobs, the date at which date the operations were started and the date at which the operations have ended.

```
SELECT ComputerName, MIN(LogTimeGenerated) AS 'Min', MAX(LogTimeGenerated) AS
'Max'
FROM MmLog_EventRecord
INNER JOIN MmGeneral_ComputerName ON MmGeneral_ComputerName.ComputerId =
MmLog_EventRecord.LogComputerId
GROUP BY ComputerName

SELECT DbCommand, MIN(DataDate) AS 'Min', MAX(DataDate) AS 'Max'
FROM MonitorCommandData
INNER JOIN MonitorCommandDefinition ON MonitorCommandDefinition.DbCommandId =
MonitorCommandData.DbCommandId
GROUP BY DbCommand

SELECT DbComputer, DbDisk_name, MIN(DataDate) AS 'Min', MAX(DataDate) AS 'Max'
FROM MonitorDiskData
INNER JOIN MonitorDiskDefinition ON MonitorDiskDefinition.DbDiskId =
MonitorDiskData.DbDiskId
GROUP BY DbComputer, DbDisk_name

SELECT DbComputer, DbObject, DbCounter, MIN(DataDate) AS 'Min', MAX(DataDate) AS
'Max'
FROM MonitorPerformance_counterData
INNER JOIN MonitorPerformance_counterDefinition ON
MonitorPerformance_counterDefinition.DbPerformance_counterId =
MonitorPerformance_counterData.DbPerformance_counterId
GROUP BY DbComputer, DbObject, DbCounter

SELECT DbObject_identifier_1, MIN(DataDate) AS 'Min', MAX(DataDate) AS 'Max'
FROM MonitorSnmp_getData
INNER JOIN MonitorSnmp_getDefinition ON MonitorSnmp_getDefinition.DbSnmp_getId =
MonitorSnmp_getData.DbSnmp_getId
GROUP BY DbObject_identifier_1

SELECT DbHost, MIN(DataDate) AS 'Min', MAX(DataDate) AS 'Max'
FROM MonitorPingData
INNER JOIN MonitorPingDefinition ON MonitorPingDefinition.DbPingId =
MonitorPingData.DbPingId
GROUP BY DbHost
```

To interpret the results, save the query output to a text file and analyze the output carefully. The contents and column values are:

Interpreting results:

1. Event log data: computer, first date of data entry, last date of data entry
2. Command monitors: command name, first date, last date
3. Disk monitors: computer, disk, first date, last date
4. Performance counters: computer, object, counter, first date, last date
5. SNMP get monitors: OID, first date, last date
6. Ping monitors: Host, first date, last date

Appendix: Database table specification

```
CREATE TABLE [dbo].[MONITORMAGIC_DATABASE_VERSION] (
    [MonitorMagicDatabaseVersionID] [int] IDENTITY (1, 1) NOT NULL ,
    [MonitorMagicDatabaseVersion] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MmGeneral_ComputerName] (
    [ComputerId] [int] IDENTITY (1, 1) NOT NULL ,
    [ComputerName] [nvarchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MmGeneral_MonitorMagicComputer] (
    [MmComputerId] [int] IDENTITY (1, 1) NOT NULL ,
    [MmComputerName] [nvarchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MmLog_Description] (
    [DescriptionId] [int] IDENTITY (1, 1) NOT NULL ,
    [Description] [nvarchar] (2048) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
    [DescriptionHash] [nvarchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MmLog_EventRecord] (
    [LogEventRecordId] [int] IDENTITY (1, 1) NOT NULL ,
    [LogTimeGenerated] [datetime] NOT NULL ,
    [LogTimeWritten] [datetime] NOT NULL ,
    [LogEventId] [int] NOT NULL ,
    [LogEventType] [int] NOT NULL ,
    [LogEventCategory] [int] NOT NULL ,
    [LogSourceId] [int] NOT NULL ,
    [LogComputerId] [int] NOT NULL ,
    [LogUserId] [int] NOT NULL ,
    [LogDescriptionId] [int] NOT NULL ,
    [LogLogId] [int] NOT NULL ,
    [LogMonitorMagicComputerId] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MmLog_LogName] (
    [LogId] [int] IDENTITY (1, 1) NOT NULL ,
    [LogName] [nvarchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MmLog_SourceName] (
    [SourceId] [int] IDENTITY (1, 1) NOT NULL ,
    [SourceName] [nvarchar] (64) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MmLog_UserName] (
    [UserId] [int] IDENTITY (1, 1) NOT NULL ,
    [UserName] [nvarchar] (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO
```

```

CREATE TABLE [dbo].[MonitorCommandData] (
  [DbCommandDataID] [int] IDENTITY (1, 1) NOT NULL ,
  [DbCommandID] [int] NOT NULL ,
  [DataDate] [datetime] NOT NULL ,
  [DataType] [int] NOT NULL ,
  [DbOutput_value__number_] [float] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MonitorCommandDefinition] (
  [DbCommandID] [int] IDENTITY (1, 1) NOT NULL ,
  [MonitorMagicServiceComputer] [nchar] (24) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [DbCommand] [nvarchar] (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbArguments] [nvarchar] (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbOutput_analysis_flags] [int] NULL ,
  [DbText_position] [int] NULL ,
  [DbReference_text_offset_position] [int] NULL ,
  [DbReference_text] [nvarchar] (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MonitorDiskData] (
  [DbDiskDataID] [int] IDENTITY (1, 1) NOT NULL ,
  [DbDiskID] [int] NOT NULL ,
  [DataDate] [datetime] NOT NULL ,
  [DataType] [int] NOT NULL ,
  [DbCapacity_MB_] [int] NOT NULL ,
  [DbFree_space__MB_] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MonitorDiskDefinition] (
  [DbDiskID] [int] IDENTITY (1, 1) NOT NULL ,
  [MonitorMagicServiceComputer] [nchar] (24) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [DbDisk_name] [nvarchar] (8) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbComputer] [nvarchar] (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MonitorPerformance_counterData] (
  [DbPerformance_counterDataID] [int] IDENTITY (1, 1) NOT NULL ,
  [DbPerformance_counterID] [int] NOT NULL ,
  [DataDate] [datetime] NOT NULL ,
  [DataType] [int] NOT NULL ,
  [DbValue] [float] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MonitorPerformance_counterDefinition] (
  [DbPerformance_counterID] [int] IDENTITY (1, 1) NOT NULL ,
  [MonitorMagicServiceComputer] [nchar] (24) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [DbComputer] [nvarchar] (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbObject] [nvarchar] (56) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbCounter] [nvarchar] (56) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbInstance] [nvarchar] (56) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbInstance_ID] [int] NULL ,
  [DbInstanceParentObjectName] [nvarchar] (56) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbInstanceParentName] [nvarchar] (56) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL ,
  [DbCounterType] [int] NULL
) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[MonitorPingData] (
  [DbPingDataID] [int] IDENTITY (1, 1) NOT NULL ,
  [DbPingID] [int] NOT NULL ,
  [DataDate] [datetime] NOT NULL ,
  [DataType] [int] NOT NULL ,
  [DbResponse_time] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MonitorPingDefinition] (
  [DbPingID] [int] IDENTITY (1, 1) NOT NULL ,
  [MonitorMagicServiceComputer] [nchar] (24) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [DbHost] [nvarchar] (128) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MonitorSnmp_getData] (
  [DbSnmp_getDataID] [int] IDENTITY (1, 1) NOT NULL ,
  [DbSnmp_getID] [int] NOT NULL ,
  [DataDate] [datetime] NOT NULL ,
  [DataType] [int] NOT NULL ,
  [DbData_calculated_value] [float] NOT NULL ,
  [DbData_calculated_interval] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[MonitorSnmp_getDefinition] (
  [DbSnmp_getID] [int] IDENTITY (1, 1) NOT NULL ,
  [MonitorMagicServiceComputer] [nchar] (24) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [DbHost_agent] [nvarchar] (96) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [DbSNMP_Community] [nvarchar] (96) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL ,
  [DbObject_identifier_1] [nvarchar] (96) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL ,
  [DbObject_identifier_2] [nvarchar] (96) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL ,
  [DbData_calculation_mode] [int] NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[MONITORMAGIC_DATABASE_VERSION] WITH NOCHECK ADD
  CONSTRAINT [PK_MmDbVer] PRIMARY KEY CLUSTERED
  (
    [MonitorMagicDatabaseVersionID]
  ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmGeneral_ComputerName] WITH NOCHECK ADD
  CONSTRAINT [PK_GENERAL_COMPUTER_NAME] PRIMARY KEY CLUSTERED
  (
    [ComputerId]
  ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmGeneral_MonitorMagicComputer] WITH NOCHECK ADD
  CONSTRAINT [PK_MM_COMPUTER_NAME] PRIMARY KEY CLUSTERED
  (
    [MmComputerId]
  ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

```



```
ALTER TABLE [dbo].[MmLog_Description] WITH NOCHECK ADD
    CONSTRAINT [PK_LOG_DESCRIPTION] PRIMARY KEY CLUSTERED
    (
        [DescriptionId]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmLog_EventRecord] WITH NOCHECK ADD
    CONSTRAINT [PK_LOG_EVENT_RECORD] PRIMARY KEY CLUSTERED
    (
        [LogEventRecordId]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmLog_LogName] WITH NOCHECK ADD
    CONSTRAINT [PK_LOG_NAME] PRIMARY KEY CLUSTERED
    (
        [LogId]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmLog_SourceName] WITH NOCHECK ADD
    CONSTRAINT [PK_LOG_SOURCE_NAME] PRIMARY KEY CLUSTERED
    (
        [SourceId]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmLog_UserName] WITH NOCHECK ADD
    CONSTRAINT [PK_LOG_USER_NAME] PRIMARY KEY CLUSTERED
    (
        [UserId]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorCommandData] WITH NOCHECK ADD
    CONSTRAINT [PK_DbCommand_DATA] PRIMARY KEY CLUSTERED
    (
        [DbCommandDataID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorCommandDefinition] WITH NOCHECK ADD
    CONSTRAINT [PK_DbCommand_DEF] PRIMARY KEY CLUSTERED
    (
        [DbCommandID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorDiskData] WITH NOCHECK ADD
    CONSTRAINT [PK_DbDisk_DATA] PRIMARY KEY CLUSTERED
    (
        [DbDiskDataID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorDiskDefinition] WITH NOCHECK ADD
    CONSTRAINT [PK_DbDisk_DEF] PRIMARY KEY CLUSTERED
    (
        [DbDiskID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[MonitorPerformance_counterData] WITH NOCHECK ADD
    CONSTRAINT [PK_DbPerformance_counter_DATA] PRIMARY KEY CLUSTERED
    (
        [DbPerformance_counterDataID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorPerformance_counterDefinition] WITH NOCHECK ADD
    CONSTRAINT [PK_DbPerformance_counter_DEF] PRIMARY KEY CLUSTERED
    (
        [DbPerformance_counterID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorPingData] WITH NOCHECK ADD
    CONSTRAINT [PK_DbPing_DATA] PRIMARY KEY CLUSTERED
    (
        [DbPingDataID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorPingDefinition] WITH NOCHECK ADD
    CONSTRAINT [PK_DbPing_DEF] PRIMARY KEY CLUSTERED
    (
        [DbPingID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorSnmpp_getData] WITH NOCHECK ADD
    CONSTRAINT [PK_DbSnmpp_get_DATA] PRIMARY KEY CLUSTERED
    (
        [DbSnmpp_getDataID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorSnmpp_getDefinition] WITH NOCHECK ADD
    CONSTRAINT [PK_DbSnmpp_get_DEF] PRIMARY KEY CLUSTERED
    (
        [DbSnmpp_getID]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmGeneral_ComputerName] ADD
    CONSTRAINT [CS_UNIQUE_GENERAL_COMPUTER_NAME] UNIQUE NONCLUSTERED
    (
        [ComputerName]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmGeneral_MonitorMagicComputer] ADD
    CONSTRAINT [CS_UNIQUE_MM_COMPUTER_NAME] UNIQUE NONCLUSTERED
    (
        [MmComputerName]
    ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [IX_EVENTREC_TIMEGEN] ON
[dbo].[MmLog_EventRecord] ([LogTimeGenerated]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [IX_EVENTREC_TIMEWRITTEN] ON
[dbo].[MmLog_EventRecord] ([LogTimeWritten]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO
```

```
CREATE INDEX [IX_EVENTREC_SRC_CAT_ID_DES] ON
[dbo].[MmLog_EventRecord] ([LogSourceId], [LogEventCategory], [LogEventId],
[LogDescriptionId]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [IX_EVENTREC_COMP_SRC_CAT_ID_DES] ON
[dbo].[MmLog_EventRecord] ([LogComputerId], [LogSourceId], [LogEventCategory],
[LogEventId], [LogDescriptionId]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [IX_EVENTREC_USER_COMP_SRC] ON
[dbo].[MmLog_EventRecord] ([LogUserId], [LogComputerId], [LogSourceId]) WITH
FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [IX_EVENTREC_DES_USER] ON
[dbo].[MmLog_EventRecord] ([LogDescriptionId], [LogUserId]) WITH FILLFACTOR = 90
ON [PRIMARY]
GO

CREATE INDEX [IX_EVENTREC_DES_COMP_USER] ON
[dbo].[MmLog_EventRecord] ([LogDescriptionId], [LogComputerId], [LogUserId]) WITH
FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [IX_EVENTREC_LOG_COMP] ON [dbo].[MmLog_EventRecord] ([LogLogId],
[LogComputerId]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmLog_LogName] ADD
CONSTRAINT [CS_UNIQUE_LOG_NAME] UNIQUE NONCLUSTERED
(
    [LogName]
) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmLog_SourceName] ADD
CONSTRAINT [CS_UNIQUE_LOG_SOURCE_NAME] UNIQUE NONCLUSTERED
(
    [SourceName]
) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MmLog_UserName] ADD
CONSTRAINT [CS_UNIQUE_LOG_USER_NAME] UNIQUE NONCLUSTERED
(
    [UserName]
) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbCommand_DATA] ON
[dbo].[MonitorCommandData] ([DbCommandID], [DataDate], [DataType]) WITH
FILLFACTOR = 90 ON [PRIMARY]
GO
```

```

ALTER TABLE [dbo].[MonitorCommandDefinition] ADD
  CONSTRAINT [UNIQUECONSTRAINT_DbCommand_DEF] UNIQUE NONCLUSTERED
  (
    [MonitorMagicServiceComputer],
    [DbCommand],
    [DbArguments],
    [DbOutput_analysis_flags],
    [DbText_position],
    [DbReference_text_offset_position],
    [DbReference_text]
  ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbCommand_DEF] ON
[dbo].[MonitorCommandDefinition] ([DbCommand], [DbArguments],
[DbOutput_analysis_flags], [DbText_position], [DbReference_text_offset_position],
[DbReference_text]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbDisk_DATA] ON [dbo].[MonitorDiskData] ([DbDiskID],
[DataDate], [DataType]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorDiskDefinition] ADD
  CONSTRAINT [UNIQUECONSTRAINT_DbDisk_DEF] UNIQUE NONCLUSTERED
  (
    [MonitorMagicServiceComputer],
    [DbDisk_name],
    [DbComputer]
  ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbDisk_DEF] ON [dbo].[MonitorDiskDefinition] ([DbDisk_name],
[DbComputer]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbPerformance_counter_DATA] ON
[dbo].[MonitorPerformance_counterData] ([DbPerformance_counterID], [DataDate],
[DataType]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorPerformance_counterDefinition] ADD
  CONSTRAINT [UNIQUECONSTRAINT_DbPerformance_counter_DEF] UNIQUE NONCLUSTERED
  (
    [MonitorMagicServiceComputer],
    [DbComputer],
    [DbObject],
    [DbCounter],
    [DbInstance],
    [DbInstance_ID],
    [DbInstanceParentObjectName],
    [DbInstanceParentName],
    [DbCounterType]
  ) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbPerformance_counter_DEF] ON
[dbo].[MonitorPerformance_counterDefinition] ([DbComputer], [DbObject],
[DbCounter], [DbInstance], [DbInstance_ID], [DbInstanceParentObjectName],
[DbInstanceParentName], [DbCounterType]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

```

```
CREATE INDEX [INDEX_DbPing_DATA] ON [dbo].[MonitorPingData] ([DbPingID],
[DataDate], [DataType]) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorPingDefinition] ADD
CONSTRAINT [UNIQUECONSTRAINT_DbPing_DEF] UNIQUE NONCLUSTERED
(
    [MonitorMagicServiceComputer],
    [DbHost]
) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbPing_DEF] ON [dbo].[MonitorPingDefinition] ([DbHost])
WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbSnmp_get_DATA] ON
[dbo].[MonitorSnmp_getData] ([DbSnmp_getID], [DataDate], [DataType]) WITH
FILLFACTOR = 90 ON [PRIMARY]
GO

ALTER TABLE [dbo].[MonitorSnmp_getDefinition] ADD
CONSTRAINT [UNIQUECONSTRAINT_DbSnmp_get_DEF] UNIQUE NONCLUSTERED
(
    [MonitorMagicServiceComputer],
    [DbHost_agent],
    [DbSNMP_Community],
    [DbObject_identifier_1],
    [DbObject_identifier_2],
    [DbData_calculation_mode]
) WITH FILLFACTOR = 90 ON [PRIMARY]
GO

CREATE INDEX [INDEX_DbSnmp_get_DEF] ON
[dbo].[MonitorSnmp_getDefinition] ([DbHost_agent], [DbSNMP_Community],
[DbObject_identifier_1], [DbObject_identifier_2], [DbData_calculation_mode])
WITH FILLFACTOR = 90 ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[MmLog_EventRecord] ADD
    CONSTRAINT [FK_LOG_EVENT_RECORD_COMPUTER_ID] FOREIGN KEY
    (
        [LogComputerId]
    ) REFERENCES [dbo].[MmGeneral_ComputerName] (
        [ComputerId]
    ),
    CONSTRAINT [FK_LOG_EVENT_RECORD_DESCRIPTION_ID] FOREIGN KEY
    (
        [LogDescriptionId]
    ) REFERENCES [dbo].[MmLog_Description] (
        [DescriptionId]
    ),
    CONSTRAINT [FK_LOG_EVENT_RECORD_LOG_ID] FOREIGN KEY
    (
        [LogLogId]
    ) REFERENCES [dbo].[MmLog_LogName] (
        [LogId]
    ),
    CONSTRAINT [FK_LOG_EVENT_RECORD_MM_COMPUTER] FOREIGN KEY
    (
        [LogMonitorMagicComputerId]
    ) REFERENCES [dbo].[MmGeneral_MonitorMagicComputer] (
        [MmComputerId]
    ),
    CONSTRAINT [FK_LOG_EVENT_RECORD_SOURCE_ID] FOREIGN KEY
    (
        [LogSourceId]
    ) REFERENCES [dbo].[MmLog_SourceName] (
        [SourceId]
    ),
    CONSTRAINT [FK_LOG_EVENT_RECORD_USER_ID] FOREIGN KEY
    (
        [LogUserId]
    ) REFERENCES [dbo].[MmLog_UserName] (
        [UserId]
    )
)
GO

ALTER TABLE [dbo].[MonitorCommandData] ADD
    CONSTRAINT [FK_ITEMID_MonitorCommandDefinition] FOREIGN KEY
    (
        [DbCommandID]
    ) REFERENCES [dbo].[MonitorCommandDefinition] (
        [DbCommandID]
    )
)
GO

ALTER TABLE [dbo].[MonitorDiskData] ADD
    CONSTRAINT [FK_ITEMID_MonitorDiskDefinition] FOREIGN KEY
    (
        [DbDiskID]
    ) REFERENCES [dbo].[MonitorDiskDefinition] (
        [DbDiskID]
    )
)
GO

ALTER TABLE [dbo].[MonitorPerformance_counterData] ADD
    CONSTRAINT [FK_ITEMID_MonitorPerformance_counterDefinition] FOREIGN KEY
    (
        [DbPerformance_counterID]
    ) REFERENCES [dbo].[MonitorPerformance_counterDefinition] (
        [DbPerformance_counterID]
    )
)
GO
```



```
ALTER TABLE [dbo].[MonitorPingData] ADD
    CONSTRAINT [FK_ITEMID_MonitorPingDefinition] FOREIGN KEY
    (
        [DbPingID]
    ) REFERENCES [dbo].[MonitorPingDefinition] (
        [DbPingID]
    )
GO

ALTER TABLE [dbo].[MonitorSnmp_getData] ADD
    CONSTRAINT [FK_ITEMID_MonitorSnmp_getDefinition] FOREIGN KEY
    (
        [DbSnmp_getID]
    ) REFERENCES [dbo].[MonitorSnmp_getDefinition] (
        [DbSnmp_getID]
    )
GO
```