Best Practices for Storing the ArcGIS® Workflow Manager Workspace in an Enterprise Geodatabase for SQL Server

An Esri® White Paper
October 2013

380 New York Street
Redlands, California 92373-8100 USA

800 447 9778
info@esri.com

esri.com

J10043

Best Practices for Storing the ArcGIS Workflow Manager Workspace in an Enterprise Geodatabase for SQL Server

An Esri White Paper

Contents **Page**

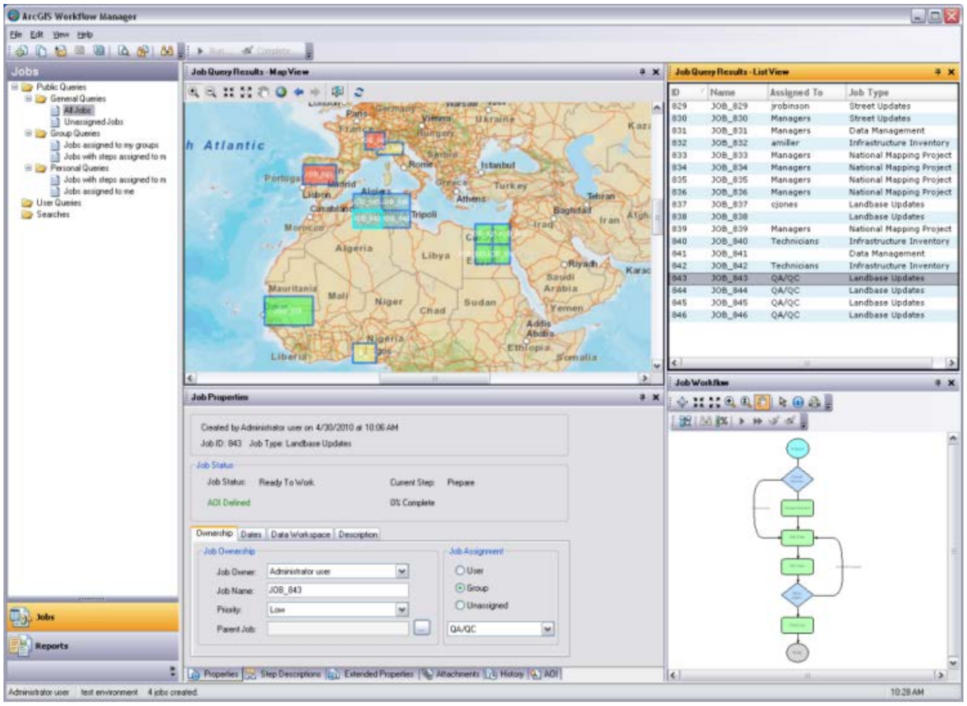
Contents **Page**

J10043

Best Practices for Storing the ArcGIS Workflow Manager Workspace in an Enterprise Geodatabase for SQL Server

Introduction

This white paper is intended to help database administrators establish the ArcGIS® Workflow Manager workspace in an enterprise geodatabase for SQL Server®. The enterprise geodatabase uses ArcSDE® technology as the gateway between geographic information system (GIS) clients and SQL Server. The Workflow Manager workspace uses ArcSDE, and this paper covers the best practices to create the ArcSDE instance. ArcGIS Workflow Manager is an enterprise workflow management application that provides an integration framework for ArcGIS multiuser geodatabase environments. It simplifies many aspects of job management and tracking and streamlines the workflow, resulting in significant time savings for any implementation. Workflow Manager provides tools for allocating resources and tracking the status and progress of jobs.



ID	Name	Assigned To	Job Type
829	JOB_829	jrobinson	Street Updates
830	JOB_830	Managers	Street Updates
831	JOB_831	Managers	Data Management
832	JOB_832	amiller	Infrastructure Inventory
833	JOB_833	Managers	National Mapping Project
834	JOB_834	Managers	National Mapping Project
835	JOB_835	Managers	National Mapping Project
836	JOB_836	Managers	National Mapping Project
837	JOB_837	cjones	Landbase Updates
838	JOB_838		Landbase Updates
839	JOB_839	Managers	National Mapping Project
840	JOB_840	Technicians	Infrastructure Inventory
841	JOB_841		Data Management
842	JOB_842	Technicians	Infrastructure Inventory
843	JOB_843	QA/QC	Landbase Updates
844	JOB_844	QA/QC	Landbase Updates
845	JOB_845	QA/QC	Landbase Updates
846	JOB_846	QA/QC	Landbase Updates

Created by Administrator user on 4/30/2010 at 10:06 AM
Job ID: 843 Job Type: Landbase Updates

Job Status
Job Status: Ready To Work
Current Step: Prepare
AOI Defined
0% Complete

Job Ownership
Job Owner: Administrator user
Job Name: JOB_843
Priority: Low
Parent Job:

Job Assignment
○ User
● Group
○ Unassigned
QA/QC

Administrator user | test environment | 4 jobs created | 10:28 AM

ArcGIS Workflow Manager Client Application Interface

A detailed history of job actions is automatically recorded for each job to give managers a complete report on how the job was completed. This information can be supplemented with comments and notes to provide even richer job documentation. Workflow Manager handles complex geodatabase tasks behind the scenes by assisting the user in the creation and management of versions. Integration between Workflow Manager and ArcGIS geodatabase tools provides a way of tracking feature edits made through Workflow Manager jobs using the geodatabase archiving tools.

Postinstallation Utility

For the Workflow Manager applications to function, all Workflow Manager system tables must exist in the database, and of those, the tables that host required configuration information must be populated with values that will drive the application behavior. The following section will detail how to create and start configuring the Workflow Manager system tables.

The Workflow Manager postinstallation utility is used to insert the Workflow Manager system tables into the workspace you've created for the Workflow Manager repository (also referred to as the Workflow Manager database).

This repository contains a set of geodatabase tables (Workflow Manager system tables) used to store the job and configuration information for your work management system and one feature class that is used to store the geometries for the areas of interest (AOIs) for your jobs.

The Workflow Manager system tables added by the postinstallation utility are geodatabase tables registered with your ArcSDE geodatabase. If at any time you are unsure of the state of the database, connect to the repository using ArcCatalog™. Because the Workflow Manager system tables are geodatabase tables, it is strongly recommended that you do not delete or modify these tables using your RDBMS tools.

Preparing the Database

Before you can install the Workflow Manager system tables into an ArcSDE instance, you must properly set up the instance to receive the repository; the steps below will show how to accomplish this task.

ArcSDE DBTUNE

DBTUNE storage parameters let you control how ArcSDE technology creates objects within a SQL Server database. You can determine such things as how to allocate space to a table or index, which Filegroup a table or index is created in, and other SQL Server-specific storage attributes. They also allow you to specify one of the available storage formats for the geometry of a spatial column.

The DBTUNE storage parameters are stored in the DBTUNE table. The DBTUNE table, along with all other metadata tables, is created in the database when the Create Enterprise Geodatabase or Enable Enterprise Geodatabase tool is executed.

When a large number of database connections access the same files in the same location on disk, database performance is slower because the connections are competing with one another for the same resources. To reduce this competition, you can store database files in different locations on disk.

Thus, DBTUNE can be modified to store the Workflow Manager system tables in separate data files across different locations on disk. This will reduce disk contention and improve database input/output (I/O).

Standard GIS storage recommendations favor keeping index and log files separate from vector and tabular business tables. For performance reasons, it is better to position the business, feature, and spatial index tables separately and position Filegroup data files based on their usage pattern. For a multiversioned, highly active editing geodatabase, database files of the VERSIONS Filegroup may be separated and dispersed across available disks to avoid I/O contention.

Disk Configuration

Large production enterprise geodatabase systems should employ a hardware striping solution. Your best disk and data organization strategies involve spreading your data across multiple disks.

With data spread across multiple disks, more spindles actively search for it. This can increase disk read time and decrease disk contention. However, too many disks can slow down a query. There are two main ways of achieving striping: Filegroups and redundant array of independent disks (RAID). You can also combine the two by creating Filegroups within disk arrays. You can employ data segregation strategies; keeping tables from indexes or certain types of tables from other tables will improve performance and alleviate administrative burdens.

The suggested SQL Server optimal configuration is as follows:

- Disk 0—SQL Server/Application software
- Disk 1—master, model, msdb
- Disk 2—tempdb
- Disk 3—Log files
- Disk 4—Feature data tables
- Disk 5—Spatial index data tables
- Disk 6—Attribute data/Business tables
- Disk 7—SQL Server indexes

Reducing Disk I/O Contention

As a rule, you should create database files as large as possible, based on the maximum amount of data you estimate the database will contain, to accommodate future growth. By creating large files, you can avoid file fragmentation and gain better database performance. In many cases, you can let data files grow automatically; just be sure to limit autogrowth by specifying a maximum growth size that leaves some hard disk space available. By putting different Filegroups on different disks, you can also minimize the physical fragmentation of your files as they grow.

To configure data and log files for best performance, follow these best practices:

- To avoid disk contention, do not put data files on the drive that contains the operating system files.

- Put transaction log files and data files on separate drives. This will give you the best performance by reducing disk contention between data and transaction log files.

- Put the tempdb database on a separate drive if possible, preferably on a RAID 10 or RAID 5 system. For environments where there is intensive use of tempdb databases, you can get better performance by putting tempdb on a separate drive, which lets SQL Server perform tempdb operations in parallel with database operations.

- The RAID configuration that is best for your database files depends on several factors, including performance and recoverability needs. RAID 10 is the recommended RAID system for transaction log, data, and index files. If you have budget restrictions, you can consider keeping the transaction log files in a RAID 10 system and storing the data and index files in a RAID 5 system.

  For more information about RAID, see RAID Levels and SQL Server at http://technet.microsoft.com/en-us/library/ms190764(SQL.105).aspx and chapter 7, "Planning Fault Tolerance and Avoidance," by Charlie Russel and Sharon Crawford, from *Microsoft® Windows® 2000 Server Administrator's Companion* (Microsoft Press) at http://technet.microsoft.com/en-us//library/bb742464(en-us).aspx.

- Use partitioning on large tables. Partitioning lets you split a table across multiple Filegroups; by using partitions, you can place a subset of a table or index on a designated Filegroup. This capability lets you separate specific pieces of a table or index onto individual Filegroups and effectively manage file I/O for volatile tables. Partitions let you easily manage archival routines and data loading operations.

Below is a suggested design to reduce disk I/O contention:

File Type	Database Activity	Move File to Disk With
Transaction log files	Frequent edits	Relatively low I/O
Transaction log files	Few or no edits	Moderate I/O
tempdb	Frequent edits	Low I/O but separate from transaction log files
master, model, msdb	Few edits	Moderate I/O
Data	Frequent edits	Relatively low I/O

Transparent Data Encryption

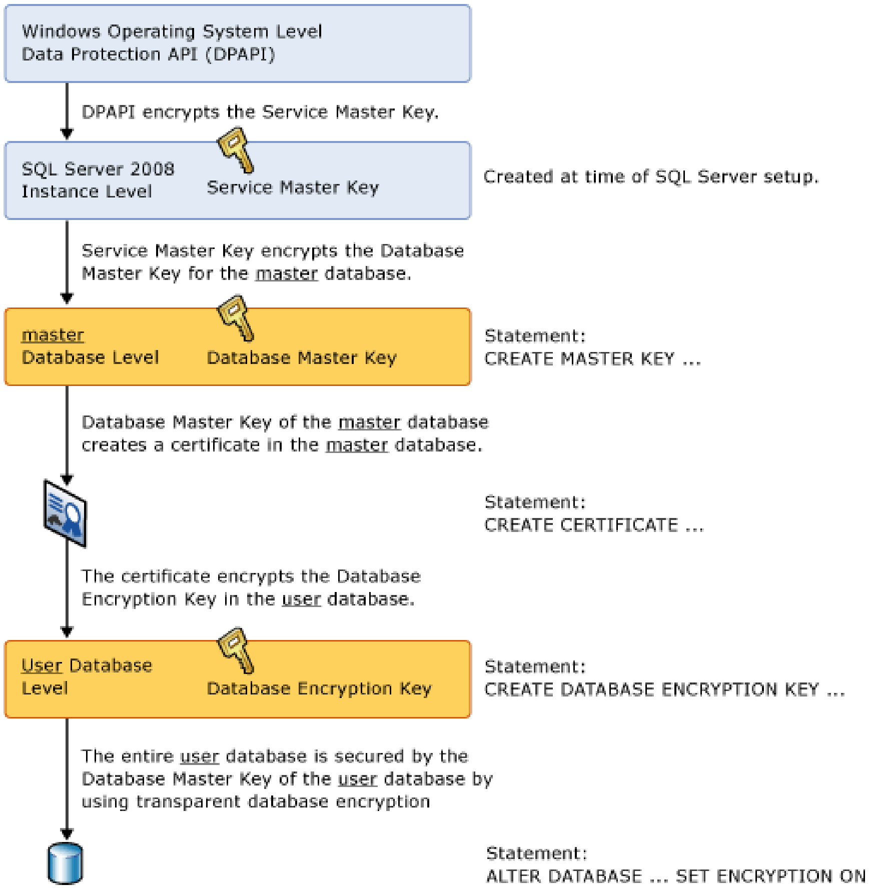
The precautions you can take to help secure the database include designing a secure system, encrypting confidential assets, and building a firewall around the database servers. However, if the physical media (drives or backup tapes) are stolen, a malicious party can just restore or attach the database and browse the data. One solution is to encrypt the sensitive data in the database and protect the keys that are used to encrypt the data with a certificate. This prevents anyone without the keys from using the data, but this kind of protection must be planned in advance.

Transparent data encryption (TDE) performs real-time I/O encryption and decryption of the data and log files. The encryption uses a database encryption key (DEK), which is stored in the database boot record for availability during recovery. The DEK is either a symmetric key secured by using a certificate stored in the master database of the server or an asymmetric key protected by an extensible key management (EKM) module. TDE protects data "at rest," meaning the data and log files. It provides the ability to comply with many laws, regulations, and guidelines established in various industries. This enables software developers to encrypt data by using Advanced Encryption Standard

(AES) and Triple Data Encryption Standard (3DES) encryption algorithms without changing existing applications.

Database files are encrypted at the page level. The pages in an encrypted database are encrypted before they are written to disk and decrypted when read into memory. TDE does not increase the size of the encrypted database.

The following illustration shows the architecture of TDE encryption:


Windows Operating System Level
Data Protection API (DPAPI)
DPAPI encrypts the Service Master Key.
SQL Server 2008
Instance Level
Service Master Key
Created at time of SQL Server setup.
Service Master Key encrypts the Database Master Key for the master database.
master
Database Level
Database Master Key
Statement:
CREATE MASTER KEY ...
Database Master Key of the master database creates a certificate in the master database.
Statement:
CREATE CERTIFICATE ...
The certificate encrypts the Database Encryption Key in the user database.
User Database
Level
Database Encryption Key
Statement:
CREATE DATABASE ENCRYPTION KEY ...
The entire user database is secured by the Database Master Key of the user database by using transparent database encryption
Statement:
ALTER DATABASE ... SET ENCRYPTION ON

TDE Encryption Architecture

Learn more about TDE at http://msdn.microsoft.com/en-us/library/bb934049.aspx.

To use TDE, follow these steps:

- Create a master key.
- Create or obtain a certificate protected by the master key.
- Create a database encryption key and protect it with the certificate.
- Set the database to use encryption.

```
USE master
GO
/* Verify master key */
SELECT * FROM sys.symmetric_keys WHERE name LIKE '%MS_DatabaseMasterKey%'
GO
/* if there are no records found, then it means there was no predefined Master Key.
 To create a Master Key, you can execute the below mentioned TSQL code. */
/* Create master key */
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'wmxadmin';
GO
/* Backup master key */
OPEN MASTER KEY DECRYPTION BY PASSWORD = 'wmxadmin';
GO
BACKUP MASTER KEY TO FILE = 'D:\mssqlbackup\master\masterkey.mk'
    ENCRYPTION BY PASSWORD = 'wmxadmin';
GO
/* Create Certificate */
CREATE CERTIFICATE wmx_cert WITH SUBJECT = 'WMX Server Certificate';
GO
/* Verify Certificate */
SELECT * FROM sys.certificates where [name] = 'wmx_cert'
GO
/* Backup certificate */
BACKUP CERTIFICATE wmx_cert TO FILE = 'D:\mssqlbackup\master\wmx.cer'
   WITH PRIVATE KEY (
         FILE = 'D:\mssqlbackup\master\wmx.pvk',
         ENCRYPTION BY PASSWORD = 'wmxadmin');
GO
USE wmxdb
GO
/* Create Encryption key */
CREATE DATABASE ENCRYPTION KEY
   WITH ALGORITHM = AES_256
   ENCRYPTION BY SERVER CERTIFICATE wmx_cert;
GO
/* Encrypt database */
ALTER DATABASE wmxdb SET ENCRYPTION ON;
GO
/* Verify Encryption */
SELECT
DB_NAME(database_id) AS DatabaseName
,Encryption_State AS EncryptionState
,key_algorithm AS Algorithm
,key_length AS KeyLength
FROM sys.dm_database_encryption_keys
GO
SELECT
NAME AS DatabaseName
,IS_ENCRYPTED AS IsEncrypted
FROM sys.databases where name ='wmxdb'
GO
```

Step 1: Create Data Files

Create new Filegroups to store the Workflow Manager system tables.

FILEGROUP	ArcSDE_PARAMETER
WMX_BDATA	Business table
WMX_BINDEX	Business table index
WMX_FDATA	Feature table
WMX_FINDEX	Feature table index
WMX_SDATA	Spatial Index table
WMX_SINDEX	Spatial Index table index
WMX_ADATA	Adds table (versioned)
WMX_AINDEX	Adds table index
WMX_DDATA	Deletes table (versioned)
WMX_DINDEX	Deletes table index

```
USE MASTER
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_BDATA]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Bdata01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Bdata01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_BDATA]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_BINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Bindex01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Bindex01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_BINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_FDATA]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Fdata01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Fdata01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_FDATA]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_FINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Findex01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Findex01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_FINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_SDATA]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Sdata01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Sdata01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_SDATA]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_SINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Sindex01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Sindex01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_SINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_ADATA]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Adata01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Adata01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_ADATA]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_AINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Aindex01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Aindex01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_AINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_DDATA]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Ddata01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Ddata01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_DDATA]
GO
ALTER DATABASE [WMXDB] ADD FILEGROUP [WMX_DINDEX]
GO
ALTER DATABASE [WMXDB] ADD FILE(NAME = N'wmx_Dindex01', FILENAME =
N'D:\mssql\data\wmxdb\wmx_Dindex01.NDF' , SIZE = 1, MAXSIZE = 400, FILEGROWTH = 1) TO FILEGROUP
[WMX_DINDEX]
GO
```

By setting the data files' initial size to 1 MB, there is no delay in the creation of the Filegroups; to avoid fragmentation, you can resize the data files.

```
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Bdata01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Bindex01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Fdata01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Findex01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Sdata01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Sindex01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Adata01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Aindex01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Ddata01', SIZE = 400MB )
ALTER DATABASE [WMXDB] MODIFY FILE ( NAME = N'wmx_Dindex01', SIZE = 400MB )
```

Verify Filegroups and data files:

```
EXEC sp_helpdb wmxdb
GO
```

Step 2: Create Workflow Manager User

Create a new database user to store the Workflow Manager system tables; grant the appropriate permissions.

Create user and schema:

```
USE WMXDB
GO
CREATE USER [wmx] FOR LOGIN [wmx]
GO
CREATE SCHEMA [wmx] AUTHORIZATION [wmx]
GO
ALTER USER [wmx] WITH DEFAULT_SCHEMA=[wmx]
GO
```

Grant privileges:

```
USE WMXDB
GO
EXEC sp_droprolemember 'gis_data_creator', 'wmx'
GO
EXEC sp_droprole 'gis_data_creator'
GO
CREATE ROLE gis_data_creator AUTHORIZATION dbo
GO
GRANT CREATE TABLE TO gis_data_creator
GO
GRANT CREATE PROCEDURE TO gis_data_creator
GO
GRANT CREATE VIEW TO gis_data_creator
GO
EXEC sp_addrolemember 'gis_data_creator', 'wmx'
GO
```

Verify role:

```
EXEC sp_helprolemember 'gis_data_creator'
GO
```

Verify role permissions:

```
select dp.NAME AS principal_name,
 dp.type_desc AS principal_type_desc,
 o.NAME AS object_name,
 p.permission_name,
 p.state_desc AS permission_state_desc
 from sys.database_permissions p
 left OUTER JOIN sys.all_objects o
 on p.major_id = o.OBJECT_ID
 inner JOIN sys.database_principals dp
 on p.grantee_principal_id = dp.principal_id
 where dp.NAME = 'gis_data_creator'
GO
```

Verify user permissions:

```
select USER_NAME(p.grantee_principal_id) AS principal_name,
 dp.type_desc AS principal_type_desc,
 p.class_desc,
 OBJECT_NAME(p.major_id) AS object_name,
 p.permission_name,
 p.state_desc AS permission_state_desc
from sys.database_permissions p
inner JOIN sys.database_principals dp
on p.grantee_principal_id = dp.principal_id
where USER_NAME(p.grantee_principal_id) = 'wmx'
```

Associate login wmx with user wmx:

```
USE WMXDB
GO
EXEC sp_change_users_login 'update_one','wmx','wmx'
GO
EXEC sp_helpuser 'wmx'
GO
```

Step 3: Modify DBTUNE

Export the dbtune file before making any modification.

```
sdedbtune -o export -f dbtune_exp.sde -u sde -p sde
-i sde:sqlserver:mcsdbsrv –D wmxdb
```

Copy ***dbtune_exp.sde*** to ***dbtune_wmx.sde***.

Create a new DBTUNE keyword following the steps below:

- Copy the ##DEFAULTS keyword and paste it at the end of the *dbtune_wmx.sde* file.

- Rename it ##WMX and change the Filegroup name for the appropriate parameters.

```
dbtune_wmx.sde

##WMX
A_INDEX_RASTER      "WITH FILLFACTOR = 75 ON WMX_AINDEX"
A_INDEX_ROWID       "WITH FILLFACTOR = 75 ON WMX_AINDEX"
A_INDEX_SHAPE       "WITH FILLFACTOR = 75 ON WMX_AINDEX"
A_INDEX_STATEID     "WITH FILLFACTOR = 75 ON WMX_AINDEX"
A_INDEX_USER        "WITH FILLFACTOR = 75 ON WMX_AINDEX"
A_INDEX_XML         "WITH FILLFACTOR = 75 ON WMX_AINDEX"
A_STORAGE           "ON WMX_ADATA"
B_INDEX_RASTER      "WITH FILLFACTOR = 75 ON WMX_BINDEX"
B_INDEX_ROWID       "WITH FILLFACTOR = 75 ON WMX_BINDEX"
B_INDEX_SHAPE       "WITH FILLFACTOR = 75 ON WMX_BINDEX"
B_INDEX_TO_DATE     "WITH FILLFACTOR = 75 ON WMX_BINDEX"
B_INDEX_USER        "WITH FILLFACTOR = 75 ON WMX_BINDEX"
B_INDEX_XML         "WITH FILLFACTOR = 75 ON WMX_BINDEX"
B_STORAGE           "ON WMX_BDATA
D_INDEX_ALL         "WITH FILLFACTOR = 75 ON WMX_DINDEX"
D_INDEX_DELETED_AT  "WITH FILLFACTOR = 75 "ON WMX_DINDEX"
D_STORAGE           "ON WMX_DDATA"
F_INDEX_AREA        "WITH FILLFACTOR = 75 ON WMX_FINDEX"
F_INDEX_FID         "WITH FILLFACTOR = 75 ON WMX_FINDEX"
F_INDEX_LEN         "WITH FILLFACTOR = 75 ON WMX_FINDEX"
F_STORAGE           "ON WMX_FDATA"
GEOMETRY_STORAGE    "GEOMETRY"
GEOMTAB_PK          "WITH FILLFACTOR = 75 ON WMX_FINDEX"
GEOMTAB_STORAGE     "ON WMX_FDATA"
I_STORAGE           "ON WMX_FDATA"
S_INDEX_ALL         "WITH FILLFACTOR = 75 ON WMX_SINDEX"
S_INDEX_SP_FID      "WITH FILLFACTOR = 75 ON WMX_SINDEX"
S_STORAGE           "ON WMX_SDATA"
END
```

If your database only stores the WMX repository, you can edit ##DEFAULTS; otherwise, create a new configuration keyword as described above.

- Import the modified ***dbtune_wmx.sde*** file.

```
sdedbtune -o import -f dbtune_wmx.sde -u sde -p sde
-i sde:sqlserver:mcsdbsrv –D wmxdb
```

Step 4: Configure SQL Server Parameters

It is recommended that you use the following parameter values when creating a SQL Server database:

SQL Server Parameters for Workflow Manager

Parameter Name	Value
Server Memory: Use AWE to allocate memory	Enabled
Processors: Boost SQL Server Priority	Enabled
Security SQL Server and Windows Authentication mode	Enabled
Connections: Maximum number of concurrent connections	0 = Unlimited
Connections: Allow remote connections to this server	Enabled

Step 5: Configure ArcSDE Parameters

You need to configure the MAXBLOBSIZE and TCPKEEPALIVE parameters for the ArcSDE geodatabase used as the WMX repository. The MAXBLOBSIZE value is -1 by default. However, if you are using SQL Server or another enterprise DBMS, make sure that this value is set to -1 and the TCPKEEPALIVE value is set to TRUE. This command should be used from the command prompt of a machine where ArcSDE is installed.

```
sdeconfig -o alter -v MAXBLOBSIZE=-1-i <service> -u sde -p <sde_password>
sdeconfig -o alter -v TCPKEEPALIVE=TRUE -i <service> -u sde -p <sde_password>
```

For more information, see the ArcSDE Administration Command Reference.

Step 6: Create and Configure the Workspace

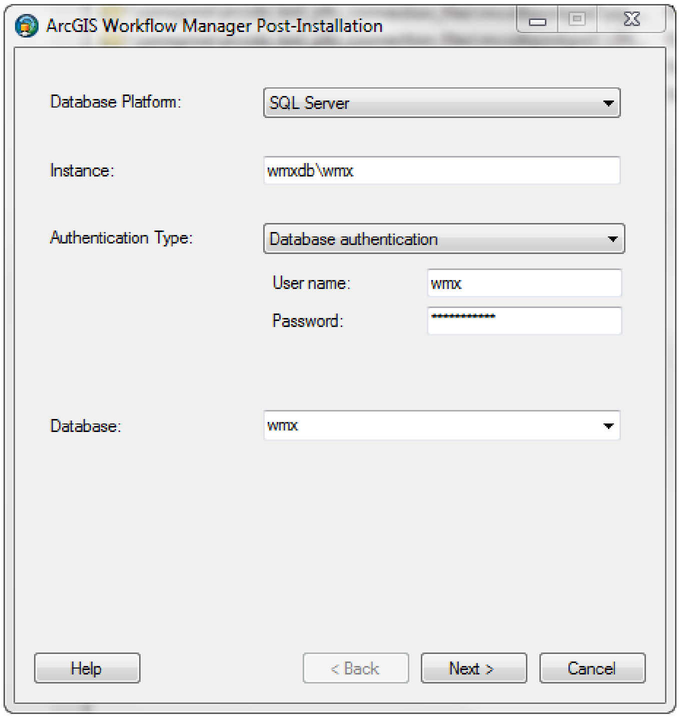
The postinstallation utility creates all the system tables required for you to use ArcGIS Workflow Manager.

Steps

a. From the **Start** menu, navigate to **ArcGIS Workflow Manager** and select **Workflow Manager Post Installation**.

   The **ArcGIS Workflow Manager Post-Installation** wizard appears.

b. Enter the connection information for the designated workspace, then test the connection information.



Database Platform: SQL Server

Instance: wmxdb\wmx

Authentication Type: Database authentication

User name: wmx

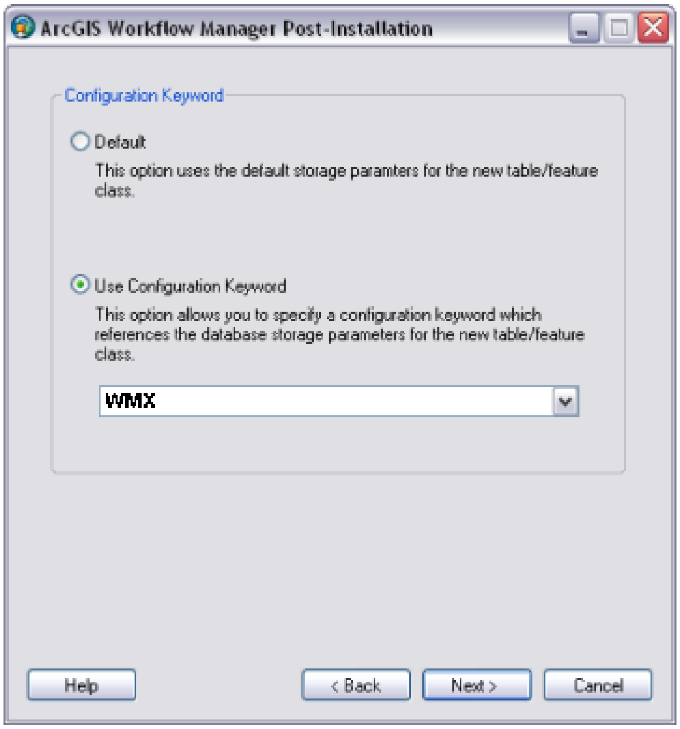
Password: ************

Database: wmx

Enter connection information for the ArcSDE geodatabase that will be the container for the Workflow Manager repository.

c. Click **Next**.

d. Choose a configuration keyword for your Workflow Manager geodatabase; **use the WMX keyword created in step 3.**



ArcGIS Workflow Manager Post-Installation

Configuration Keyword

○ Default

This option uses the default storage paramters for the new table/feature class.

● Use Configuration Keyword

This option allows you to specify a configuration keyword which references the database storage parameters for the new table/feature class.

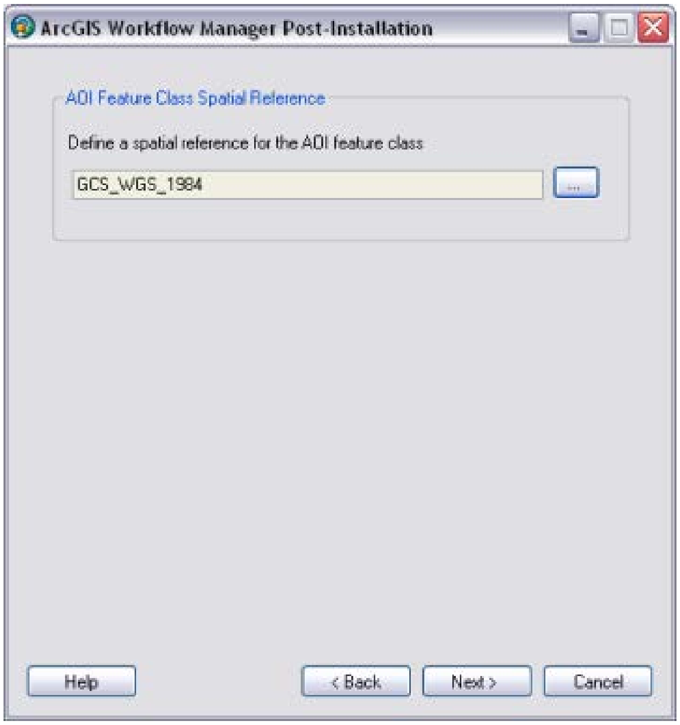
WMX

Choose the configuration keyword.

e. Click **Next**.

f. Define the spatial reference for the Area of Interest feature class.

A separate two-page wizard appears allowing you to select the coordinate system and extents.



ArcGIS Workflow Manager Post-Installation

AOI Feature Class Spatial Reference

Define a spatial reference for the AOI feature class

GCS_WGS_1984

Define the spatial reference for the Area of Interest feature class.

g. Click **Next**.

Choose your starting configuration option for the Workflow Manager repository.

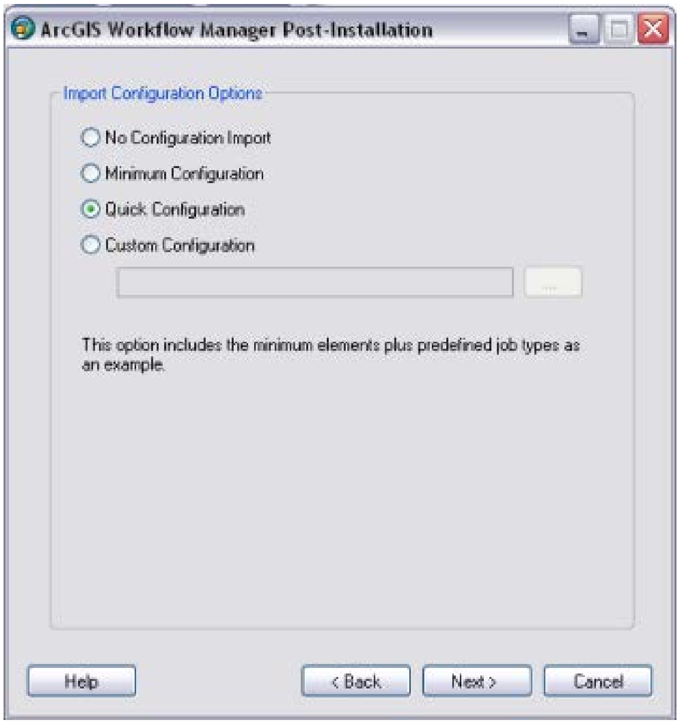
- **No configuration** will put the Workflow Manager system tables schema into your designated geodatabase and will not prepopulate any of the configuration elements.

- **Minimum configuration** imports just the basic elements that the Workflow Manager system requires.

  Note: If you plan to use all the functionality available in the Workflow Manager client application, it is recommended that you import at least the minimum configuration.

- **Quick configuration** includes the minimum elements plus predefined job types as an example.

- **Custom configuration** allows you to import a preexisting Workflow Manager configuration file.


ArcGIS Workflow Manager Post-Installation
Import Configuration Options
○ No Configuration Import
○ Minimum Configuration
● Quick Configuration
○ Custom Configuration
...
Help
< Back
Next >
Cancel

This option includes the minimum elements plus predefined job types as an example.

Choose the starting configuration for your repository.

Note: If you plan to import a Workflow Manager configuration exported from a 3.x database, choose the **No Configuration Import** option. Tools are provided in the Workflow Manager Configuration Manager to import the legacy configuration files.

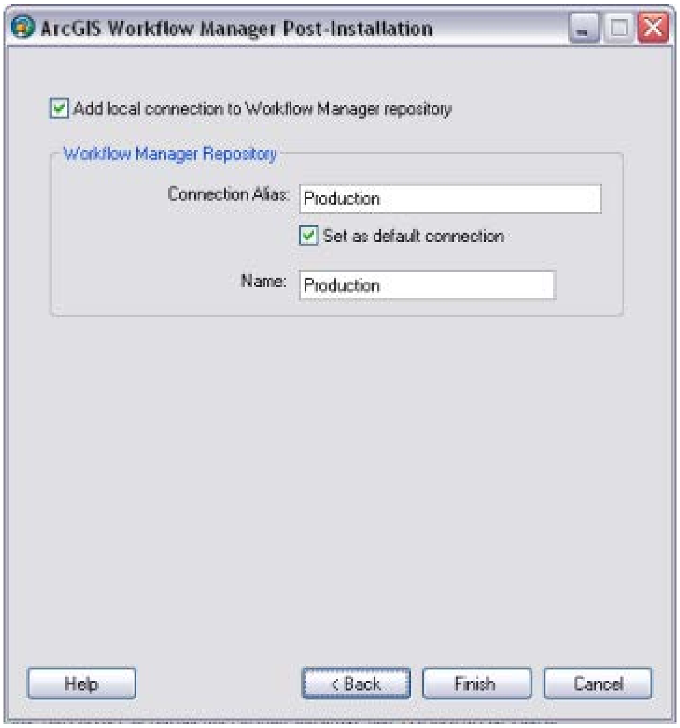
h. Click **Next**.

i. Choose whether you would like to create a Workflow Manager database connection to this new repository when the postinstallation is complete.

- Specify a connection alias.
- Specify a repository name.

  Note: This is used as a unique identifier when defining a cluster for Repository Replication.

- Specify to set the database as default.



ArcGIS Workflow Manager Post-Installation

☑ Add local connection to Workflow Manager repository

Workflow Manager Repository

Connection Alias: Production

☑ Set as default connection

Name: Production

Decide if you want Workflow Manager to add a Workflow Manager database connection for you.

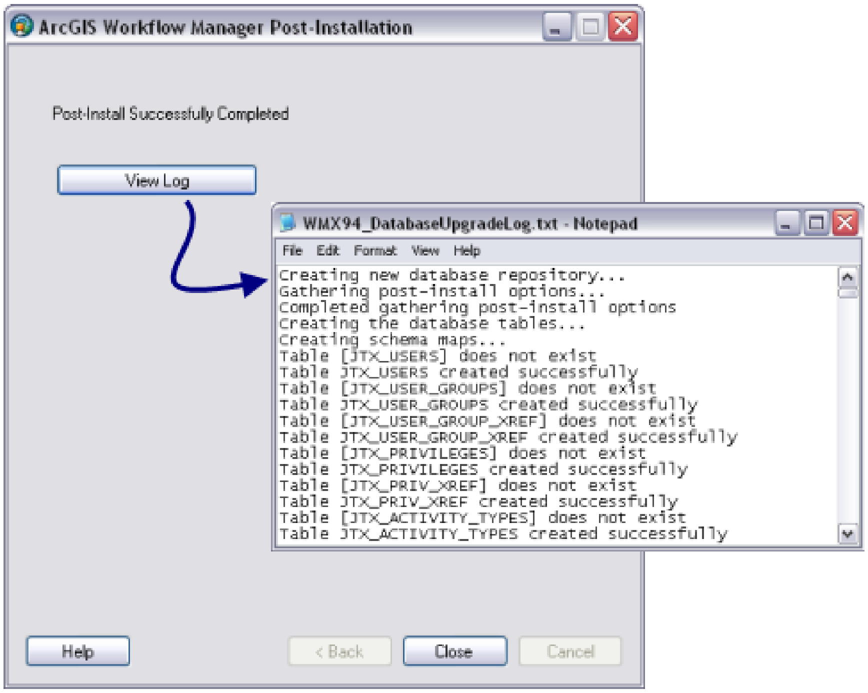
j. Click **Finish**.

The system tables needed to use Workflow Manager are created in the database. Also, the current user running the postinstallation is automatically added to the list of users and has administrator access.

The final page in the wizard appears, indicating whether the postinstallation was successful.

Note: Administrator access means that the current user can connect to the selected database through the configuration manager to make edits to components of your system.

k. Choose whether you would like to view the log.


ArcGIS Workflow Manager Post-Installation
View Log
WMX94_DatabaseUpgradeLog.txt - Notepad
File Edit Format View Help
Help
< Back
Close
Cancel

Post-Install Successfully Completed

```
Creating new database repository...
Gathering post-install options...
Completed gathering post-install options
Creating the database tables...
Creating schema maps...
Table [JTX_USERS] does not exist
Table JTX_USERS created successfully
Table [JTX_USER_GROUPS] does not exist
Table JTX_USER_GROUPS created successfully
Table [JTX_USER_GROUP_XREF] does not exist
Table JTX_USER_GROUP_XREF created successfully
Table [JTX_PRIVILEGES] does not exist
Table JTX_PRIVILEGES created successfully
Table [JTX_PRIV_XREF] does not exist
Table JTX_PRIV_XREF created successfully
Table [JTX_ACTIVITY_TYPES] does not exist
Table JTX_ACTIVITY_TYPES created successfully
```

The final page of the wizard indicates the success of the postinstallation and provides an option to see a log of the process.

l. Click Close.

Step 7: Verify Storage

Run the SQL queries below to verify that the Workflow Manager workspace was created under the correct Filegroups.

```
USE WMXDB
GO
```

List Filegroups and data files:

```
EXEC sp_helpdb wmxdb
GO
```

List Filegroup data files:

```
EXEC sp_helpfilegroup 'PRIMARY'
GO
```

List tables by Filegroup:

```
SELECT USER_NAME(o.uid) [Owner],
 OBJECT_NAME(i.id) [Table Name],
 FILEGROUP_NAME(groupid) AS [Filegroup Name]
 FROM sysindexes i inner join sysobjects o
 ON i.id = o.id
 WHERE i.indid IN (0, 1) AND OBJECTPROPERTY(i.id, 'IsMSShipped') = 0 AND
 USER_NAME(o.uid) = 'wmx'
 ORDER BY 1,3,2
GO
```

List indexes by table and Filegroup:

```
select 'owner'=user_name(o.uid)
,'table_name'=object_name(i.id),i.indid
,'index_name'=i.name ,i.groupid
,'filegroup'=f.name ,'file_name'=d.physical_name
,'dataspace'=s.name from sys.sysindexes i
,sys.sysobjects o,sys.filegroups f
,sys.database_files d, sys.data_spaces s
where objectproperty(i.id,'IsUserTable') = 1
and i.id = o.id
and f.data_space_id = i.groupid
and f.data_space_id = d.data_space_id
and f.data_space_id = s.data_space_id
and user_name(o.uid) = 'wmx'
order by object_name(i.id),i.name,f.name
go
```

If any tables or indexes are stored in the wrong Filegroup, use ALTER TABLE and ALTER INDEX to change the Filegroup (see SQL Server Books Online at http://msdn.microsoft.com/en-us/library/ms130214.aspx).

Also, in Management Studio, you can re-create the DDL script of tables and indexes. Then within *create script*, you can modify the Filegroup parameter and re-create the tables and indexes in the correct Filegroups. This is particularly useful when tables are empty and you are allowed to re-create database objects.

Step 8: Using Data Compression

Row and page compression for tables and indexes enables you to save storage space by reducing the size of the database. Data compression has the drawback of increasing CPU usage because the data must be compressed and decompressed when being accessed. You cannot use data compression with system tables, and only the Enterprise and Developer editions of SQL Server 2012 support data compression.

You can configure data compression on the following:

- Clustered tables
- Heap tables (A heap is a table without a clustered index.)
- Nonclustered indexes
- Indexed views
- Individual partitions of a partitioned table or index

There are three forms of data compression you can use with SQL Server 2012: row-level compression, Unicode compression, and page-level compression.

Note: You can learn more about heaps at http://msdn.microsoft.com/en-us/library/hh213609(v=SQL.110).aspx.

Row-Level Compression

Row-level compression works by using more efficient storage formats for fixed-length data.

Row-level compression uses the following strategies to save space:

- Storing fixed-length numeric data types and CHAR data types as though they were variable-length data types

- Not storing NULL or 0 values

- Reducing metadata required to store data

Although it does reduce the amount of space that data uses, row-level compression does not provide the storage improvements of page-level compression. The advantage of row-level compression is that it requires less CPU usage than page-level compression. You use the following syntax to compress a table by using row-level compression:

```
ALTER TABLE tableName REBUILD WITH (DATA_COMPRESSION=ROW)
```

For example, to rebuild all partitions of the wmx.TableA table of the WMXDB database by using row compression, use the following query:

```
USE [WMXDB]
ALTER TABLE [wmx].[TableA] REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = ROW)
```

You use the following syntax to configure an index with row-level compression:

```
ALTER INDEX indexName ON tableName REBUILD PARTITION ALL WITH (DATA_COMPRESSION=ROW)
```

Note: You can learn more about row-level compression at http://msdn.microsoft.com/en-us/library/cc280576(v=sql.110).aspx.

Unicode Compression

Unicode compression enables the database engine to compress Unicode values stored in page- or row-compressed objects. You can use Unicode compression with the fixed-length nchar(n) and nvarchar(n) data types. Unicode compression is automatically used where appropriate when you enable row and page compression.

Note: You can learn more about unicode compression at http://msdn.microsoft.com/en-us/library/ee240835(SQL.110).aspx.

Page-Level Compression

Page-level compression compresses data by storing repeating values and common prefixes only once and then making references to those values from other locations within the table. When page compression is applied to a table, row compression techniques are also applied. Page-level compression uses the following strategies:

- Row-level compression is applied to maximize the number of rows stored on a page.

- Column prefix compression is applied by replacing repeating data patterns with references.

- This data is stored in the page header.

- Dictionary compression scans for repeating values and then stores this information in the page header.

The benefits of page compression depend on the type of data compressed. Data that involves many repeating values will be more compressed than data populated by more unique values. You use the following general syntax to apply page-level compression:

```
ALTER TABLE name REBUILD WITH (DATA_COMPRESSION=PAGE)
```

For example, to rebuild all partitions of the wmx.TableA table of the WMXDB database by using page compression, use the following query:

```
USE [WMXDB]
ALTER TABLE [wmx].[TableA] REBUILD PARTITION = ALL
WITH
(DATA_COMPRESSION = PAGE)
```

You use the following syntax to configure an index with page-level compression:

```
ALTER INDEX indexName ON tableName REBUILD PARTITION ALL WITH (DATA_COMPRESSION=PAGE)
```

Note: You can learn more about page-level compression at http://msdn.microsoft.com/en-us/library/cc280464(v=sql.110).aspx.

If tables or indexes are partitioned, you can configure compression on a per-partition basis. If you split a partition by using the ALTER PARTITION statement, the new partitions inherit the data compression attribute of the original partition. If you merge two partitions, the resultant partition has the compression attribute of the destination partition. Although compression does allow more rows to be stored on a page, it doesn't alter the maximum row size of a table or index. You can't enable a table for compression if the maximum row size and the compression overhead exceed 8,060 bytes. The default compression setting for indexes is NONE, and you must specify the compression property for indexes when you create them. Nonclustered indexes do not inherit the compression property of the table, but clustered indexes created on a heap inherit the compression state of the heap. Data compression applies only at the source, so when you export data from a compressed source, SQL Server will output the data in uncompressed row format. Importing uncompressed data into a target table enabled for compression will compress the data.

Note: You can learn more about data compression at http://msdn.microsoft.com/en-us/library/cc280449(v=sql.110).aspx.

You can configure compression by using the preceding Transact-SQL statements or from SQL Server Management Studio by using the Data Compression Wizard on either tables or indexes. You can use the Data Compression Wizard to add and remove compression.

To use the Data Compression Wizard to change the compression settings for both tables and indexes, perform the following steps:

a. In SQL Server Management Studio, right-click the table or index you want to compress, choose **Storage**, and then select **Manage Compression**.

b. On the **Welcome To The Data Compression Wizard** page, click **Next**.

c. On the **Select Compression Type** page, you can choose to use the same compression type for all partitions or choose among **Row**, **Page**, and **None** on a per-partition basis. Click **Calculate** to determine the difference between current space usage and compressed usage.

d. On the **Select An Output Option** page, choose whether to create a script, to perform the operation immediately, or to perform the option according to a schedule. Click **Next** and then click **Finish** to complete the wizard.

Note: You can learn more about the Data Compression Wizard at http://msdn.microsoft.com/en-us/library/cc280496(v=SQL.110).aspx.

Estimating Compression

The best way to determine the benefits of compression on an object is to use the sp_estimate_data_compression_savings stored procedure. The benefits of compression depend on factors such as the uniqueness of data. The sp_estimate_data_compression_savings stored procedure is available in the Enterprise edition of SQL Server only.

The syntax of the stored procedure is as follows:

```
sp_estimate_data_compression_savings[ @schema_name = ] 'schema_name', [ @object_name = ]
'object_name', [@index_id = ] index_id,[@partition_number = ] partition_number,
[@data_compression = ] 'data_compression'
```

For example, to configure an estimate of the compression benefits of using row compression on the wmx.TableA table in the WMXDB database, execute the following Transact-SQL statement:

```
USE WMXDB;
GO
EXEC sp_estimate_data_compression_savings 'wmx', 'TableA', NULL, NULL,
'ROW';
GO
```

To configure an estimate of the compression benefits of using page compression on the same table, execute the following Transact-SQL statement:

```
USE WMXDB;
GO
EXEC sp_estimate_data_compression_savings 'wmx', 'TableA', NULL, NULL,
'PAGE';
GO
```

Note: You can learn more about how to estimate compression savings at http://msdn.microsoft.com/en-us/library/cc280574(v=sql.110).aspx.

You can use the SQL script below to list the WMX user tables and indexes and to generate the SQL statements to set page compression for the tables and indexes.

```
/*--------------------------------------------------------------------
-- Verify WMX Schema Storage
--------------------------------------------------------------------*/
USE [wmxdb]
GO
/*--------------------------------------------------------------------
--List all tables
--------------------------------------------------------------------*/
SELECT USER_NAME(o.uid) [owner], o.name,o.id,o.type,o.status
FROM sysobjects o
WHERE USER_NAME(o.uid) = 'wmx'
AND o.type <> 'S' and o.type = 'U'
ORDER BY o.name,o.type;
GO
/*--------------------------------------------------------------------
--List all indexes
--------------------------------------------------------------------*/
SELECT USER_NAME(o.uid) [owner], OBJECT_NAME(i.id) [table], i.name [index],o.type [type]
FROM sysindexes i inner join sysobjects o ON i.id = o.id
WHERE USER_NAME(o.uid) = 'wmx'
AND o.type <> 'S' and o.type = 'U' and i.indid = 1
ORDER BY USER_NAME(o.uid),OBJECT_NAME(i.id),i.name;
GO
/*--------------------------------------------------------------------
--Table page compression
--Example:
/*
ALTER TABLE WMX.JTX_JOBS_AOI
REBUILD WITH (DATA_COMPRESSION = PAGE);
GO
*/
--------------------------------------------------------------------*/
--Generate script to set table page compression:
SELECT 'ALTER TABLE ' + USER_NAME(o.uid) + '.' + o.name + ' REBUILD WITH (DATA_COMPRESSION =
PAGE);' [TXTSQL]
FROM sysobjects o
WHERE USER_NAME(o.uid) = 'wmx'
AND o.type <> 'S' and o.type = 'U'
ORDER BY o.name,o.type;
GO
/*--------------------------------------------------------------------
--Index page compression
--Example:
/*
ALTER INDEX R172_pk
ON WMX.JTX_JOBS_AOI
REBUILD WITH ( DATA_COMPRESSION = PAGE ) ;
GO
*/
--------------------------------------------------------------------*/
--Generate script to set index page compression:
SELECT 'ALTER INDEX ' + i.name + ' ON ' + USER_NAME(o.uid) + '.' + OBJECT_NAME(i.id) +
       ' REBUILD WITH ( DATA_COMPRESSION = PAGE );' [TXTSQL]
FROM sysindexes i inner join sysobjects o ON i.id = o.id
WHERE USER_NAME(o.uid) = 'wmx'
AND o.type <> 'S' and o.type = 'U' and i.indid = 1
ORDER BY USER_NAME(o.uid),OBJECT_NAME(i.id),i.name;
GO
```

Compression and TDE

Encryption of the database file is performed at the page level. The pages are encrypted before they are written to disk and decrypted when read into memory. TDE does not increase the size of the encrypted database.

SQL Server 2012—Transparent Data Encryption (TDE)

Encrypted data compresses significantly less than equivalent unencrypted data.

If TDE is used to encrypt a database, backup compression will not be able to significantly compress the backup storage.

SQL Server 2012—Data Compression

Step 9: Grant Permissions and Roles

Grant permissions to the Workflow Manager system tables through the schema.

```
USE [wmxdb]
GO
EXEC sp_droprolemember 'wmx_editor', 'giseditor'
GO
EXEC sp_droprole 'wmx_editor'
GO
EXEC sp_addrole 'wmx_editor', 'wmx'
GO
GRANT DELETE ON SCHEMA::[wmx] TO [wmx_editor]
GRANT EXECUTE ON SCHEMA::[wmx] TO [wmx_editor]
GRANT INSERT ON SCHEMA::[wmx] TO [wmx_editor]
GRANT SELECT ON SCHEMA::[wmx] TO [wmx_editor]
GRANT UPDATE ON SCHEMA::[wmx] TO [wmx_editor]
GO
EXEC sp_droprolemember 'wmx_viewer', 'gisviewer'
GO
EXEC sp_droprole 'wmx_viewer'
GO
EXEC sp_addrole 'wmx_viewer', 'wmx'
GO
GRANT SELECT ON SCHEMA::[wmx] TO [wmx_viewer]
GO
```

Verify role:

```
EXEC sp_helprolemember wmx_editor'
GO
EXEC sp_helprolemember wmx_viewer'
GO
```

Verify role permissions:

```
select dp.NAME AS principal_name,
 dp.type_desc AS principal_type_desc,
 o.NAME AS object_name,
 p.permission_name,
 p.state_desc AS permission_state_desc
 from sys.database_permissions p
 left OUTER JOIN sys.all_objects o
 on p.major_id = o.OBJECT_ID
 inner JOIN sys.database_principals dp
 on p.grantee_principal_id = dp.principal_id
 where dp.NAME in ('wmx_editor','wmx_viewer')
GO
```

Step 10: Log File Tables

Enterprise geodatabases use log file tables to maintain lists of selected records. Records are written to log file tables for later use by the application whenever a selection of a specific size is made, a reconciliation or post on a versioned database is performed, or a disconnected editing checkout is done in a client application. The log file tables store the ObjectIDs of the selected features so they can be redisplayed. This allows faster analysis and processing of information.

In ArcGIS, by default, log file tables are used if the selection set contains 100 or more records. This selection threshold of 100 features is set in the registry. It can be changed; however, Esri does not recommend doing so. There is no proven performance reason for changing it, and doing so could even cause performance problems. Thus, log file tables store feature selections in ArcMap™ that have more than 100 records for each connected ArcSDE editor/viewer user.

Log file options are set using specific parameters in the SERVER_CONFIG and DBTUNE tables (sde_server_config and sde_dbtune in a SQL Server database). Parameters in these tables are altered using the sdeconfig and sdedbtune commands, respectively.

In SQL Server, one table is created in tempdb in the format ##SDE_SESSION<SDE_ID>. This table is truncated when the connecting application deletes its log files, and the table is dropped when the session disconnects. When using the default setting, users do not require CREATE TABLE permission in the database for the session table to be created in tempdb.

The DBTUNE SESSION_TEMP_TABLE parameter must be set to 1 (true) to allow the session-based log file table to be created in tempdb. If you change the SESSION_TEMP_TABLE parameter to 0 (false), the SDE_LOGFILES, SDE_LOGFILE_DATA, and SDE_SESSION<SDE_ID> tables will be created in the connecting user's schema; hence, the user requires CREATE TABLE permission.

Learn more about ArcSDE log file tables at resources.arcgis.com/en/help/main/10.2/index.html#/What_are_ArcSDE_log_file_tables/002q00000011000000/.

Step 11: Create Database Users

The example below shows how to create an editor and viewer ArcSDE user:

Editor User

```
USE master
GO
EXEC sp_addlogin N'giseditor', 'gis$editor', @logindb, @loginlang
GO
```

Create user:

```
USE [wmxdb]
GO
CREATE USER [giseditor] FOR LOGIN [giseditor]
GO
```

Grant privileges:

```
USE [wmxdb]
GO
EXEC sp_addrolemember N'wmx_editor', N'giseditor'
GO
```

Viewer User

```
USE master
GO
EXEC sp_addlogin N'gisviewer', 'gis$viewer', @logindb, @loginlang
GO
```

Create user:

```
USE [wmxdb]
GO
CREATE USER [gisviewer] FOR LOGIN [gisviewer]
GO
```

Grant privileges:

```
USE [wmxdb]
GO
EXEC sp_addrolemember N'wmx_viewer', N'gisviewer'
GO
```

Replication

ArcGIS Workflow Manager repository replication is designed to allow organizations to maintain multiple Workflow Manager repositories locally in different geographic regions for performance reasons, while the users at each location continue to work on the same set of jobs.

You will define a collection of Workflow Manager repositories that you want to have identical Workflow Manager contents. This collection of repositories is known as a repository cluster.

One Workflow Manager Repository is designated as the parent repository and is tasked with coordinating the synchronization between each repository. After the parent repository has been specified, each child repository is added. The child repositories can be designated as either connected repositories, where the synchronization will happen automatically, or offline repositories, where the synchronization occurs by creating files that can be manually imported.

Learn more about replicating Workflow Manager repositories at resources.arcgis.com/en/help/main/10.2/index.html#/Replicating_Workflow_Manager_Repositories/003800000077000000/.

Conclusion

You can reduce disk contention and improve database I/O by storing the ArcGIS Workflow Manager workspace in different locations on disk. However, this practice alone does not guarantee optimal database performance, and additional tuning tasks may be needed.

Learn more about the recommended tuning tasks:
resources.arcgis.com/en/help/main/10.2/index.html#/Minimize_disk_I_O_contention_in_SQL_Server/002q00000021000000/

For more information on ArcGIS Workflow Manager, visit esri.com/workflowmanager or e-mail workflowmanager@esri.com.

Access blogs, forums, downloads, and more, from the ArcGIS Workflow Manager resource center at resources.arcgis.com/en/communities/workflow-manager/.

You can access other resources at ArcGIS 10.2 for Desktop Help (resources.arcgis.com/en/help/main/10.2/index.html#/Welcome_to_the_ArcGIS_Help_Library/00qn0000001p000000/) and Esri Support (support.esri.com).

Esri inspires and enables people to positively impact their future through a deeper, geographic understanding of the changing world around them.

Governments, industry leaders, academics, and nongovernmental organizations trust us to connect them with the analytic knowledge they need to make the critical decisions that shape the planet. For more than 40 years, Esri has cultivated collaborative relationships with partners who share our commitment to solving earth's most pressing challenges with geographic expertise and rational resolve. Today, we believe that geography is at the heart of a more resilient and sustainable future. Creating responsible products and solutions drives our passion for improving quality of life everywhere.

Contact Esri

380 New York Street
Redlands, California 92373-8100 USA

1 800 447 9778
T 909 793 2853
F 909 793 5953
info@esri.com
esri.com

Offices worldwide
esri.com/locations

Printed in USA