[TechNet Home](http://www.microsoft.com/technet/default.mspx) > [Products and Technologies](http://www.microsoft.com/technet/prodtechnol/default.mspx) > [SQL TechCenter Home](http://www.microsoft.com/technet/prodtechnol/sql/default.mspx) > [SQL Server 2000](http://www.microsoft.com/technet/prodtechnol/sql/2000/default.mspx) > [Maintain](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/default.mspx)

**Microsoft SQL Server 2000 Analysis Services Operations Guide**

Updated: February 25, 2004

By Carl Rabeler and Dave Wickert

**Published:** November 2003

**Summary:** This paper describes techniques you can use to operate and maintain a Microsoft� SQL Server� 2000 Analysis Services data warehouse.

**On This Page**

|  | [Introduction](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EWAA) |
| --- | --- |
|  | [Configuration Management](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EVAA) |
|  | [Release Management](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EUAA) |
|  | [Change Management](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#ETAA) |
|  | [Security Administration](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#ESAA) |
|  | [Service and Availability Management](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#ERAA) |
|  | [Capacity Management](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EQAA) |
|  | [Problem and Incident Management](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EPAA) |
|  | [Appendix A: Checklist for Operations](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EOAA) |
|  | [Appendix B: Resources](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#ENAA) |
|  | [Appendix C: How to Tune the Process Buffer Size](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EMAA) |
|  | [Appendix D: Sample Script for Changing the Data Folder Location](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#ELAA) |
|  | [Appendix E: Sample Script for Creating Repository Audit Triggers](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EKAA) |
|  | [Appendix F: Sample Script for Creating an OLAP Linked Server](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EJAA) |
|  | [Appendix G: Sample Script To Verify Analysis Services Availability](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EIAA) |
|  | [Appendix H: Sample Scriptto Determine When Lazy Processing is Complete](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EHAA) |
|  | [Appendix I: Sample Script to Determine Whether Data Slices Have Been Set](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EGAA) |
|  | [Appendix J: Sample Script to Determine the Analysis Services Edition](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EFAA) |
|  | [Appendix K: Data Folder Structure](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EEAA) |
|  | [Database Folder](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#EDAA) |
|  | [Cube Folder](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#ECAA) |

**Introduction**

Every administrator who operates the Microsoft� SQL Server� 2000 Analysis Services portion of a data warehouse faces certain common operational issues. Analysis Services and its environment must be configured appropriately. The Analysis Services application must be deployed from the development environment to the production environment. Change control must be employed to ensure that changes to an existing environment are fully tested, and that approved changes are deployed properly. Capacity issues must be anticipated proactively. Problems must be resolved quickly and consistently. The agreed-upon availability of the Analysis Services cubes for querying must be ensured.

This paper provides guidance to assist administrators with the operation and maintenance of their Analysis Services databases as components within an existing IT and database infrastructure. You should integrate existing structures and techniques to solve Analysis Services operational issues, rather than inventing new processes just for Analysis Services. For example, you should employ the same incident tracking and problem resolution techniques that you use with your relational databases; use the same automation techniques to schedule jobs and scripts; and employ the same change control techniques to ensure change is controlled, tested, and documented.

The guidance provided by this paper is presented within the structure of the Microsoft Operations Framework (MOF) methodology. MOF is a representation of the cyclical process that any operation goes through, which is divided into four quadrants: changing, operating, supporting, and optimizing. This paper addresses the changing, operating, and supporting quadrants in the following sections:

| � | "Configuration Management" discusses best practices for configuring Analysis Services and the Windows operating system on the Analysis Services computer. |
| --- | --- |
| � | "Release Management" discusses the tools you can use to move an Analysis Services database from the development environment to the quality assurance (QA) and production environments. |
| � | "Change Management" discusses the importance of controlling and managing change, and covers how to use triggers to detect unmanaged changes. |
| � | "Security Administration" discusses how to secure access to Analysis Services while minimizing overhead, and how to configure service accounts for appropriate access to Analysis Services data, the Analysis Services repository, and relational data. |
| � | "Service and Availability Management" discusses how to provide service continuity through the implementation of an availability plan that incorporates regular backup operations, well-tested restoration techniques, and clustering when continuous, 24-hour per day operations are required. |
| � | "Capacity Management" discusses memory, disk, and processor capacity issues. |
| � | "Problem and Incident Management" discusses techniques that you can employ to detect, resolve, and document Analysis Services problems and incidents. |

For information on MOF's fourth quadrant, optimizing, go to the Microsoft Technet Web site (<http://www.microsoft.com/technet/>) and see "Microsoft Analysis Services Performance Guide."

The guidance presented here is based on the collective experience of the Microsoft Business Intelligence (BI) Practices team and the Analysis Services development team. In addition to the techniques discussed in this paper, you should also apply SQL Server 2000 Service Pack 3 (SP3) to the computer on which Analysis Services is running (the Analysis server) and update Microsoft PivotTable� Service (PTS) on each Analysis Services client computer (run Ptslite.exe in the ..\Msolap\install\pts folder for SP3). Updating PTS on each client computer is particularly important because the client-server architecture of PTS places a significant portion of the PTS code on each client computer and SP3 includes significant performance and security enhancements to the client-side components of PTS. For information on determining the level of service pack that has been applied to an Analysis Services installation or to a client computer, see "Verifying the Appropriate Service Pack Level" later in this paper. For information on automating the installation of SP3 on each client computer, go to Microsoft Knowledge Base (support.microsoft.com) and see the article "Ptssetup.exe Sample Automatically Downloads and Installs OLAP Client."

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Configuration Management**

Before you release and deploy your BI application, you must install and configure the appropriate edition of the Windows operating system and Analysis Services on the computer that will host the application. You should maintain a record of the actual configuration of this computer, the operating system, and the installed services and applications in a run book. Having a written record containing this configuration information will help you rebuild the server in the event of a disaster. You can also refer to this written record whenever you need to obtain configuration information for troubleshooting. This run book should also contain information about systems from which data is received, and contact information about people who should be contacted in the event of an emergency. For information about the detailed information that you should keep in your run book, go to the Microsoft Technet Web site (<http://www.microsoft.com/technet/>) and see "Appendix: Contents of a Run Book" in *Microsoft SQL Server 2000 High Availability Series, Volume 1: Planning Guide*. While the topic list in the *Planning Guide* is primarily oriented to the RDBMS components of Microsoft SQL Server 2000, it does include topics specifically for SQL Server Analysis Services. In addition, there are many points in common, such as resource and contact information, details on the hardware configuration, and some operational and emergency tasks.

After you release and deploy your BI application, there are a number of configuration settings that you need to monitor to see if they require modification as conditions change. If you change any configuration settings, you must update the information in the run book to ensure that all members of your administration team can quickly determine the current configuration of Analysis Services and the Microsoft Windows� operating system. For more information on managing change, see "Change Management" later in this paper.

You should also document all of the objects in Analysis Services. For example, you can use OLAP Scribe, which is a Microsoft Word template that allows you to generate complete documentation of Analysis Services using Decision Support Objects (DSO). You can obtain this template at <http://go.microsoft.com/fwlink/?LinkId=22012>.

**Configuring the Windows Operating System**

Configuring the Windows operating system for optimum Analysis Services performance consists primarily of configuring processors, the Windows paging files, and memory. You can also disable services that are not needed.

**Note**   Microsoft recommends that Analysis Services not be installed on a domain controller. There are situations where this is required, such as when Analysis Services is installed with Microsoft Small Business Server or on a standalone domain, but, in general, you should avoid the configuration if possible. For more information, go to Microsoft Knowledge Base (support.microsoft.com) and see the article "INF: Running OLAP Services on a Domain Controller."

**Processor**

If you are running Analysis Services on a multiple processor computer, Analysis Services schedules threads on all available processors in the computer. Unlike the SQL Server service, Analysis Services does not natively support processor affinity to control the processors on which its threads will execute. Because Analysis Services is highly multithreaded, Analysis Services can consume all available processing resources. For this reason, you should use a dedicated server for Analysis Services in most cases. If you must share the computer resources with other server applications, you should select a server application that supports processor affinity, such as SQL Server. By setting processor affinity in SQL Server, you can control the processors executing the SQL Server threads and the priority of these threads, to ensure that sufficient processor resources remain available for Analysis Services threads.

If you need to control the processors on which Analysis Services threads execute, you should also consider using Microsoft Windows Server� 2003 Enterprise Edition or Windows Server 2003 Datacenter Edition. These editions of Windows Server 2003 include the Windows System Resource Manager (WSRM), which allows an administrator to set processor and memory allocation policies for applications running on the server. WSRM enables you to select the Analysis Services process and limit Analysis Services threads to specific CPUs or to a specific threshold of processor resources. For more information on WSRM, go to the Windows Server 2003 Web site (<http://www.microsoft.com/windowsserver2003/techinfo/overview/wsrmfastfacts.mspx>) and see the article "Windows System Resource Manager-Fast Facts."

**Paging Files**

By default, Windows uses a single paging file equal to approximately 1.5 times the amount of physical memory in your computer. However, because Analysis Services makes extensive use of Windows paging files, you should always add a second paging file equal to the amount of physical memory on your computer. The SQL Server relational and multidimensional runtime engines work with memory very differently. The SQL Server relational engine directly maps and controls physical memory usage, while the Analysis Services multidimensional engine relies on the Windows operating system to allocate additional memory (physical or virtual) to the Analysis Services address space as needed. As a result, when the Windows operating system reduces the Analysis Services working set because other applications require allocations of physical memory, Analysis Services may need to use the paging file for its memory needs. You must ensure that the total paging file space is more than that configured by default, so that Analysis Services has sufficient virtual memory if the Windows operating system has insufficient physical memory.

While the Windows operating system has provisions for effectively controlling the general use of memory, Microsoft strongly recommends that customers configure servers with an adequate amount of memory so that extensive paging does not occur. If the main processing component of Analysis Services, the msmdsrv process, causes extensive paging, processing performance suffers.

**Memory**

Processes (such as Analysis Services) running in Windows 2000 Server or Windows Server 2003 Standard Edition can address a maximum of 2 gigabytes (GB) of RAM in the main process space. If you are working with large or complex cubes, Analysis Services may require more than 2 GB to load dimensions into memory, process dimensions, load replica dimensions, and still have sufficient memory for an effective query results cache. To allow Analysis Services to address more than 2 GB of RAM in a single process, you must install Windows 2000 Advanced Server; Windows 2000 Datacenter Server; Windows Server 2003 Enterprise Edition; or Windows Server 2003 Datacenter Edition.

Windows Server 2003 Enterprise Edition and Windows Server 2003 Datacenter Edition, are available in 32-bit and 64-bit versions. The 64-bit version supports the 64-bit version of Analysis Services. Because Windows 2000 Advanced Server and Windows 2000 Datacenter Server are 32-bit operating systems, only the 32-bit version of Analysis Services can be installed.

| � | The 64-bit version of Analysis Services can address all available memory in the main process space without any special configuration (up to 64 GB with the Enterprise Edition and up to 512 GB with the Datacenter Edition). |
| --- | --- |
| � | The 32-bit version of Analysis Services can address up to 3 GB of memory in the main process space, if you enable Application Memory Tuning. Unless you enable Application Memory Tuning, no process can address more than 2 GB in the main process space. To enable Application Memory Tuning on the Analysis Services computer, set the **/3 GB** switch in the boot.ini file and then use Analysis Manager to set an appropriate **Memory conservation threshold** value for Analysis Services. If you set the **/3GB** switch in boot.ini, the computer on which Analysis Services is running should have at least 4 GB of memory to ensure that the Windows operating system has sufficient memory for system services. If you are running other applications on the same computer, you must factor in their memory requirements as well. For example, if the SQL Server service and Analysis Services are installed on the same computer, SQL Server can address memory above 4 GB because SQL Server supports Address Windowing Extensions (AWE). In this case, you could install and use 8 GB or more on the server. However, because Analysis Services does not support AWE, Analysis Services cannot access more the 3 GB of memory in the main process space unless the 64-bit version is used. |

For more information on setting the **/3GB** switch, go to Microsoft Knowledge Base (support.microsoft.com) and see the article "INF: How to Enable Analysis Server To Use 3 GB of RAM.". For more information on setting the **Memory conservation threshold** value, see "Configuring Analysis Services" immediately following this section.

**Disabling Unnecessary Services**

While there is no complete list of the Windows services that are not required on an Analysis Services computer, turning off services that are not required will save memory for use by Analysis Services. Among the services that you might not need are the following:

| � | Alerter |
| --- | --- |
| � | Application Management Transfer Service |
| � | ClipBook |
| � | COM+ Event System |
| � | Computer Browser |
| � | Distributed Link Tracking Client |
| � | Distributed Transaction Coordinator |
| � | Fax Service |
| � | Indexing Service |
| � | Internet Connection Sharing |
| � | Logical Disk Manager Administrative Service |
| � | Messenger |
| � | Net Logon - This service is needed if your users require Windows NT� pass-through authentication to connect to Analysis Services. |
| � | Microsoft NetMeeting� Remote Desktop Sharing |
| � | Network DDE |
| � | Network DDE DSDM |
| � | NT LM Security Support Provider |
| � | Performance Logs and Alerts |
| � | Protected Storage |
| � | QoS RSVP |
| � | Remote Access Auto Connection Manager |
| � | Remote Access Connection Manager |
| � | Remote Procedure Call (RPC) Locator |
| � | Routing and Remote Access |
| � | RunAs Service |
| � | Security Accounts manager |
| � | Server |
| � | SmartCard |
| � | SmartCard Helper |
| � | System Event Notification |
| � | Task Scheduler |
| � | TCP/IP NetBIOS Helper Service |
| � | Telephony |
| � | Telnet |
| � | Uninterruptible Power Supply |
| � | Utility Manager |
| � | Windows Installer |
| � | Windows Time |

You can turn off a service by either disabling the service or setting the service to start manually. If you set a service to start manually, Windows starts the service if it is needed.

**Note**   Viruses can also start services that are set to manual.

If you disable the service, Windows cannot start the service. For a complete listing of Windows 2000 services and their functions, go to the Microsoft Windows 2000 Web site (<http://www.microsoft.com/windows2000/>) and see the article "Glossary of Windows 2000 Services."

**Note**   If you are running SQL Server 7.0, do not disable the remote registry service. This service is required to administer a remote Analysis Services installation.

**Important**   You should disable the Indexing Service to avoid locking problems and possible corruption during processing. You should also configure any anti-virus software on the machine so it does not scan the Analysis Services Data folder or the Temporary file folder. To locate these folders using Analysis Manager, right-click the server, select **Properties**, and then view the information that appears on the **General** tab.

**Configuring Analysis Services**

After you install Analysis Services, there are a number of configuration settings that you should always check (several of which you should generally change). Most configuration settings can be changed using Analysis Manager, although there are several settings that you must change by editing the Windows registry directly. This paper discusses the most important configuration settings. For information on additional configuration settings related to performance, go to the Technet Web site (<http://www.microsoft.com/technet/>) and see "Microsoft Analysis Services Performance Guide." For information about the entire range of registry entries for Analysis Services, go to the Microsoft MSDN� library (msdn.microsoft.com) and see "Registry Entries for Microsoft SQL Server 2000 Analysis Services." If you want to monitor the registry keys that are written to by Analysis Services (or by any other Windows application), you can use various utilities that are available for the Windows platform. A commonly used utility is Regmon from Sysinternals (<http://www.sysinternals.com/>). This freeware registry monitoring utility shows the applications that are accessing the registry, the keys they are accessing, and the registry data they are reading and writing.

**Note**   Some features of Analysis Services are only available with the Enterprise Edition. Use the sample DSO script provided in Appendix J, "Sample Script to Determine the Analysis Services Edition," later in this paper to determine the edition of Analysis Services that you are using.

**Memory Settings**

Having sufficient memory for Analysis Services increases query responsiveness and processing performance. Properly configuring available memory will maximize the use of memory, limit the use of disk resources for processing, and prevent the cleaner thread from evicting cache entries too quickly. The amount of memory used by Analysis Services for various purposes is regulated by a number of memory settings:

| � | High and low memory settings |
| --- | --- |
| � | Very Large Dimension Memory (VLDM) threshold setting |
| � | Process buffer settings |

These settings are configured using default values or based on the amount of physical memory in the computer during installation. Changing some of these memory settings is generally recommended.

**High and Low Memory Settings**

Analysis Services employs a number of mechanisms to keep the amount of memory allocated to it within the range that is defined by two settings on the **Environment** tab of the **Server Properties** dialog box in Analysis Manager: the **Memory conservation threshold** and the **Minimum allocated memory** settings (the **HighMemoryLimit** and the **LowMemoryLimit** values in the registry). The default value for the **Memory conservation threshold** setting is the amount of physical memory on the computer at the time of installation. The default value for the **Minimum allocated memory** setting is half the amount of physical memory on the computer at the time of installation. If you change the amount of memory on the computer after installation, you must manually modify these values. Otherwise, Analysis Services will not properly utilize the actual amount of physical memory on the computer.

Analysis Services uses a cleaner thread to reduce the amount of memory allocated to Analysis Services when the amount of memory allocated reaches the halfway point between the **Memory conservation threshold** setting and the **Minimum allocated memory** setting. When the cleaner thread is activated, it begins evicting entries in the query results cache, based on a cost/benefit algorithm that takes into account a variety of factors, including how frequently the data in the query results cache is being used, the amount of resources that were required to resolve the entries, and the amount of space being consumed by related entries. By default, the cleaner thread runs at below-normal priority. The frequency with which it runs is determined by the **BackgroundInterval** registry setting. The default value is thirty seconds. This setting actually governs the number of seconds between processing periods for a variety of background tasks, including the cleaner thread, query logging, and lazy processing. If you want to set an interval for the cleaner thread separate from these other background tasks, add the **CleanerInterval** registry key and set a value just for the cleaner thread.

When the amount of memory used by Analysis Services exceeds the **Memory conservation threshold** setting, Analysis Services increases the priority of the cleaner thread to normal in order to quickly reduce the allocated memory to the **Minimum allocated memory** setting. If the total memory allocated to all Analysis Services tasks exceeds the memory conservation threshold by more than approximately 6.25 percent, Analysis Services immediately begins dropping the cache entries for entire cubes in order to quickly reduce the amount memory used by Analysis Services. In this scenario, because Analysis Services is shedding memory extremely quickly, the total amount of allocated memory may drop below the **Minimum allocated memory** setting.

If you set the Minimum allocated memory setting too low, the cleaner thread removes too many cached entries from the query results cache. This reduces query response times and requires additional resources to repopulate the query results cache. For example, suppose your computer has 2 GB of physical memory and you set the **Memory conservation threshold** setting to 1.4GB and the **Minimum allocated memory** setting to 100MB. If memory usage ever goes significantly above 1.4 GB, the cleaner thread aggressively drops entries from the query results cache, down to 100 megabytes (MB) or even less. Analysis Services must then rebuild the entries in the query results cache from newly submitted queries. A more appropriate **Minimum allocated memory** setting for this system is approximately 1 GB, which gives the cleaner thread room to perform its job without unnecessarily throwing away cache entries when the amount of allocated memory exceeds the **Memory conservation threshold** setting.

As you can see, setting the **Memory conservation threshold** setting too low will also reduce overall performance, and may result in out-of-memory errors. You should never set the **Memory conservation threshold** setting to more than the amount of physical memory on the computer (otherwise the paging files will be used excessively). If you enable the **/3 GB** switch, you should not set the **Memory conservation threshold** setting to more than approximately 2.7 GB. Setting this value slightly below the 3-GB memory limit ensures that the cleaner thread has sufficient time to respond to low memory conditions and to reduce allocated memory before Analysis Services uses he entire 3-GB address space. The memory conservation threshold does not directly limit the amount of memory used by Analysis Services, which means Analysis Services can run out of address space in the main process space or use more memory than is physically present on the computer.

**Tip**   If you add memory or enable the /3 GB switch in the boot.ini file, increase the Memory conservation threshold and Minimum allocated memory settings in Analysis Manager.

**Note**   If you have not installed Analysis Services Service Pack 3, you must modify the **HighMemoryLimit** value by editing the registry directly to enable Analysis Services to address more than 2 GB of memory rather than using the **Memory conservation threshold** setting in Analysis Manager. In SP3, Analysis Manager was changed to allow an administrator to enter a number larger than 2 GB (up to 3 GB). In SP2 and earlier, Analysis Manager would only allow settings between 1 and 2047 MB. For more information, go to Microsoft Knowledge Base (support.microsoft.com) and see the article "INF: How to Enable Analysis Server To Use 3 GB of RAM."

**Very Large Dimension Memory (VLDM) Threshold**

The 32-bit version of Analysis Services (the 64-bit version does not use VLDM) attempts to prevent large dimensions from using all of the available virtual memory address space by loading each very large dimension at startup into its own process space with its own virtual memory address space. A very large dimension is one that exceeds the value of the **VLDMThreshold** setting in the registry. The default VLDM threshold is 64 MB. While using a separate address space for each dimension that exceeds the VLDM threshold does save virtual memory address space for other uses in the main process, overall performance slows when one or more dimensions exceed the VLDM threshold. Loading all dimensions into the main process space (when possible) yields better performance, but you must ensure that there is sufficient virtual memory address space to perform the following:

| � | Load all dimensions into memory at startup. |
| --- | --- |
| � | Load all dimensions being processed in parallel or in a single transaction into memory during processing (these are called shadow dimensions). Analysis Services uses the existing version of each dimension to resolve user queries until the processing transaction commits. To minimize the amount of memory needed for shadow dimensions, process dimensions in separate transactions. If you select **Process the Database** or **Process All Dimensions** in Analysis Manager, the dimensions are processed in a single transaction and will require sufficient memory to load each dimension in memory twice (once at startup and then again during processing). |
| � | Store replica dimensions as required. See "Replica Dimensions" later in this paper. |
| � | Perform all processing without using temporary files. See "Process Buffer" later in this paper. However do not use VLDM just to allow a larger process buffer. Processing is a one-time or, at worst, an occasional activity. Using VLDM is a constant overhead for performance and complexity (more processes, more context switching, and so on). |
| � | Create and use a sufficiently large query results cache. See "Query Results Cache" later in this paper. |

If Analysis Services does not have enough virtual memory address space in the main process space, set the VLDM threshold so that only the largest dimensions are loaded into separate address spaces. For more information on how Analysis Services uses available memory and how to calculate the amount of memory required, see "Capacity Management" later in this paper.

If Analysis Services has sufficient virtual memory address space in the main process space, disable VLDM by renaming the msmdvldm.exe file in the Bin folder to some other file name (such as msmdvldm-disabled.exe). When the service starts, if it can't find the VLDM executable, the service disables it. VLDM is automatically disabled on a 64-bit system. Disabling VLDM ensures that all dimensions are loaded into the main process space.

With all of the performance and restrictions associated with VLDM, the usual best practice is that if your application is large enough to be forced into using VLDM, then you should evaluate whether SQL Server 2000 (64-bit) will provide better performance.

**Important**In general, you should consider using the 64-bit version of Analysis Services if:

| � | Your BI application includes large dimensions or many member properties. |
| --- | --- |
| � | There are many databases and cubes in the same Analysis Services instance. |
| � | You must support a high level of querying while simultaneously processing partitions. |
| � | You are unable to increase the size of the process buffer to eliminate the use of temporary files on disk during processing. |

For more information, go to the Technet Web site (<http://www.microsoft.com/technet/>) and see "Microsoft SQL Server 2000 (64-bit) Analysis Services: Why Migrate, and What to Expect If You Do."

**Note**   When the VLDM threshold is used for large dimensions, Analysis Services creates the shadow dimensions for these large dimensions during processing in the main address space. Thus, even when you are using VLDM, there is still a considerable impact on the virtual address space of the main process.

**Process Buffer**

Analysis Services creates a process buffer in memory for each partition it processes. It allocates memory to each buffer as it is needed, and releases this memory from each buffer when partition processing is complete. Analysis Services uses each buffer for two separate tasks.

| � | First, Analysis Services loads fact data for the partition from the read-ahead buffer into the process buffer and then sorts, indexes, and writes the fact data into the fact level of the MOLAP partition file in segments. The sorting process includes as much data as can be held in the process buffer at one time. |
| --- | --- |
| � | Second, Analysis Services uses the process buffer for calculating aggregations. If the process buffer in memory is not large enough to hold all calculations during this phase, Analysis Services supplements the process buffer with temporary files on disk to complete the calculation of aggregations. |

The **Process buffer size** setting on the **Processing** tab in Analysis Manager (the **ProcessReadSegmentSize** value in the registry) determines the maximum size of each process buffer. By default, the maximum size of each process buffer is approximately 32 MB. For most applications, this is probably too small and should be immediately increased. A more effective setting is at least 150 to 200 MB.

If the size of each process buffer is large enough to efficiently sort and index large portions of the fact data before it is written to segments in the partition file, overall data organization and query responsiveness improve. Furthermore, if the fact table contains many duplicate records, a large process buffer allows Analysis Services to merge duplicate records in memory, saving space and improving query performance.

If Analysis Services exceeds the size of a process buffer while creating aggregations, Analysis Services changes its algorithm to use temporary files that augment the memory allocated to the process buffer. If temporary files are used, Analysis Services moves aggregations between these temporary files and the memory allocated to the process buffer as the aggregations are being calculated. Reading and writing these temporary files is much slower than in-memory calculations and very I/O-intensive. You should tune your system to eliminate the use of these temporary files by increasing the **Process buffer size** setting when possible. All aggregations for a partition are calculated at once and must fit into memory; otherwise temporary files are used.

When processing multiple partitions in parallel or processing an entire cube in a single transaction, you must ensure that the total memory required for the process buffers, dimensions, shadow dimensions, replicas, and other memory requirements do not exceed the **Memory conservation threshold** setting. If Analysis Services runs out of virtual address space for these simultaneous operations, you receive an out-of-memory error. If you have insufficient physical memory to back the virtual memory, the Windows operating system uses the virtual memory paging files to supplement the available physical memory. While the use of the paging files has performance implications if excessive paging occurs, a small amount of paging (approximately 100 to 200 MB) is generally acceptable if necessary.

On the other hand, if the process buffer setting is too large and if the number and size of the aggregates is large enough to fill the process buffer during processing, Analysis Services may exceed the memory conservation threshold (which causes the query response cache to be trimmed or dumped). Exceeding the memory conservation threshold during processing causes temporary files to start being used. Remember that if you are processing partitions in parallel, each partition uses a separate process buffer.

**Tip**   If you have sufficient memory, increase the Process Buffer Size setting to at least 150 - 200 MB to eliminate the use of temporary files during processing. It is not uncommon to set the process buffer size to 300 or 500 MB on servers with large cubes. To determine an appropriate process buffer size, follow the procedure in Appendix C, "How to Tune the Process Buffer Size," later in this paper.

**Data and Temporary File Locations**

An Analysis Services instance has a Data folder and a Temporary folder. Analysis Services uses the Data folder to store the multidimensional structures for all the objects defined on the Analysis Services instance. It uses the Temporary folder to supplement the memory allocated to each process buffer when a process buffer is too small for the aggregations being processed. The default location for both of these folders is C:\Program Files\Microsoft Analysis Services\Data. You can change the location for either or both during setup or after installation. To change the location after setup, right-click the Analysis server object in Analysis Manager, and then click **Properties**. You can also use the sample script provided in Appendix D, "Sample Script for Changing the Data Folder Location, to change the Data folder programmatically.

**Note**If you use virus-scanning software on the Analysis Services computer, you should disable scanning of the Analysis Services Data, Temporary, and Bin folders.

You should place the Data folder on its own RAID array; RAID 10 or RAID 1 + 0 provides the best performance but RAID 5 is frequently fast enough for many Analysis Services installations. The main activity of Analysis Services is reading data from the files in the Data folder in response to user queries, not writing to files in the Data folder. Once you determine the amount of space required for the data, index, and aggregation structures, you should allocate approximately double that amount of disk space to allow sufficient space to enable you to refresh the data and hold shadow files during processing. For more information on calculating the amount of space required for the data folder, see "Disk" in the "Capacity Management" section later in this paper. For information about each type of file stored in the Data folder, see Appendix K, "Data Folder Structure."

**Note**   Because the Data folder stores security files that control end users' access to Analysis Services objects, you must secure the Data folder against unauthorized access. Only members of the OLAP Administrators group and the Administrators group should have access to the Data folder. If you move the Data folder location after installation, you must configure these security settings manually. For more information on securing Analysis Services, see "Security Administration" later in this paper.

**Temporary Folder**

You should place the Temporary folder, if it is actually used, on a RAID array that yields excellent write performance and that is on a different physical drive than the Data folder. Consider using RAID 0, 1, 0+1, or 10 depending on your budget requirements and amount of use. However, for best performance, it is more important to allocate a sufficiently large process buffer to obviate the need for temporary files during processing. If processing requires temporary files, the algorithm is an order of magnitude slower than if the process buffer was large enough to perform the processing entirely in memory. If you find that the files in the Temporary folder structure are used extensively and you cannot eliminate their use, youcan add a second Temporary file folder on a different physical drive by adding the **TempDirectory2** registry key (HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\OLAP Server\Current Version) and specifying a location on a separate physical drive for the second Temporary folder. When you must use temporary files, using two Temporary folders increases processing performance because the data in one Temporary folder is sequentially read, merged with new segment data, and then written to the second Temporary folder (in 64-kilobyte segments). The data in the second Temporary folder is then read, merged with new segment data, and written to the first Temporary folder. This process continues until the calculation of aggregations is complete. To determine whether the Temporary folder is being used, see Appendix C, "How to Tune the Process Buffer Size" later in this paper.

**Data Source Configuration**

When you create a new database within an Analysis Services instance, one of your first tasks is to define the data source for the database. A data source contains the information necessary to access source data for the database objects. The term "data source" actually refers to the data source object that is created, not the source data itself. When you define the data source in Analysis Manager, the name given to the object is either *<server\_name>*-*<database\_name>* or *<localhost>*-*<database\_name>*. However, to eliminate confusion when the database is moved to another server, you should change the default naming convention by creating a logical name for the data source unrelated to the name of the original computer on which the database was initially created.

To create a logical name for the data source object in Analysis Manager, create the data source object. Then copy the new data source object and paste it into the same Analysis Services database. You are then prompted to define a new name for the data source object. The name you choose should reflect the logical type of data such as Sales Data, or Personal Data. After you define the new logical name, delete the original data source object. Thereafter, when you move an Analysis Services database between computers, you can simply change the underlying server and database in the connection string by modifying the properties of the data source object in Analysis Manager (or in your script).

In addition to renaming your data sources to logical rather than physical names, you should ensure that your deployment computers use the same name. If your development computer has its data source named Sales Data, then your QA computer should have its data source named Sales Data, and your production computer should have its data source named Sales Data. Using consistent names across the development, QA, and production computers makes migration of individual pieces easier by cutting and pasting between the Analysis Services databases.

If you do not change the name of the data source object before you create objects in the database, you will not be able to change the name of the data source object without using third-party utilities. For more information on tools you can use when moving a database between Analysis Services instances, see "Release Management" later in this paper.

**Service Accounts**

To understand the permissions required for the MSSQLServerOLAPService and SQL Server Agent service accounts, you need to understand the security context in which various operations are executed. Certain tasks are performed in the context of the logged-on user, and other tasks are performed n the security context of the MSSQLServerOLAPService service account.

| � | When you request that Analysis Manager create a new object or browse for an existing object, the task is executed in the security context of the interactive user account of the user performing the task. |
| --- | --- |
| � | When Analysis Services processes dimensions, partitions, and mining models, this task is performed in the security context of the MSSQLServerOLAPService service account. You must grant sufficient rights to this service account to ensure that processing will succeed. |

It is not uncommon for users to assume that if they can create objects, then they can process them. While this is frequently the case in a simple one-computer development environment, in a multiple-computer production department you're likely to encounter problems. With source databases residing on separate servers from the Analysis Services database, and the Analysis Services instance being managed remotely, frequently the first problem you experience when you roll out such an application into production is insufficient permissions.

You must ensure that the MSSQLServerOLAPService and the SQL Server Agent service accounts have sufficient permissions for the tasks they need to perform. At a minimum, the service account must be a member of the OLAP Administrators group. This permission is needed for any user (or service that is acting on behalf of a user) that manages an Analysis Services server.

**MSSQLServerOLAPService**

By default, the MSSQLServerOLAPService service runs under the local system account, which has full administrator permissions on the local computer, but no access rights to remote computers. Because the local system account has no network access, in many cases you will need to change the service account to an account that can be granted network access rights. With Windows 2000, this will be a domain user account. If you are using Windows Server 2003, you can use the NetworkService account.

The account under which the MSSQLServerOLAPService service runs must have sufficient permissions to perform several different tasks. The MSSQLServerOLAPService service must be able to process Analysis Services objects, access source data during processing, and receive security credentials in a multiple-tier environment.

**Tip**   With the named pipes network protocol, the process (such as an Analysis Services processing operation) attempting to access an application (such as SQL Server) on a remote computer must be authenticated by the Windows operating system before the process can be authenticated by SQL Server. With TCP/IP sockets, the process does not normally have to be authenticated by the Windows operating system before presenting its authentication credentials to SQL Server.

**SQL Server Agent**

When you automate object creation tasks (such as for creating partitions) or object processing using SQL Server Agent, the service account used by the SQL Server Agent service must have sufficient rights to create or process the object.

**Processing**

When you run the MSSQLServerOLAPService service under a domain user account (or the NetworkService account), you must add this account to the OLAP Administrators local group on the Analysis Services computer to enable Analysis Services to process dimensions, partitions, and mining roles on that computer. Membership by this account in the OLAP Administrators group allows the MSSQLServerOLAPService service to access the registry, the Data folder, and the Temporary folder. User accounts that are not members of the OLAP Administrators group should not have access to these locations.

**Source Data Access**

The MSSQLServerOLAPService service account must also have logon account permissions to access the source data in the source database if trusted connections are used to access the source data. With trusted connections, the MSSQLServerOLAPService service account is used to connect to the data source. If trusted connections are not used, you can specify a user name and password.

The permissions required when connecting to a data source also depends on the type of storage structure used for the Analysis Services partitions. When MOLAP storage is used, the service account must have at least SELECT permissions on the source database. If ROLAP or HOLAP storage is used, the service account must have at least SELECT and CREATE TABLE permissions on the source database.

There are also permissions needed if using resources from other Analysis Services servers. If using linked cubes, then the MSSQLServerOLAPService service account must have read access on the cube and cell-level security must not be defined. If Analysis Services is a remote partition publisher, then the MSSQLServerOLAPService service account must be in the OLAP Administrators group on the machine where the remote partition is located. If Analysis Services is a remote partition subscriber, then the MSSQLServerOLAPService service account must have read access on the cube.

**Receive Client Security Credentials Via Middle-Tier Application**

The MSSQLServerOLAPService service account is irrelevant in the typical client-server environment. In this environment, the user application connects directly to the Analysis Services computer to execute a query or create an object, and passes the user's credentials directly to Analysis Services for evaluation. Access is granted or denied by Analysis Services based on cell-level and dimension-level security.

However, if the client application attempts to connect to Analysis Services through a middle-tier server, the authentication process is not quite so simple. Normally, security credentials cannot be passed over multiple computers. However, if the middle-tier application server and the Analysis Services computer support Kerberos authentication and delegation, the client's security credentials can be passed by the middle-tier application to Analysis Services.

For Kerberos authentication, delegation, impersonation, and mutual authentication to work, the MSSQLServerOLAPService service must run under one of the following types of accounts:

| � | Local system account (which has no network access rights). |
| --- | --- |
| � | Domain administrator account. |
| � | Domain user without administrative privileges in the Microsoft Active Directory domain, provided that a domain administrator registers the Service Principal Name (SPN) for the account separately using the **setspn** utility in the Windows 2000 Resource Kit.  **Note**   There are a number of steps you must follow to permit Kerberos authentication, delegation, impersonation, and mutual authentication to work. For information about these steps, see "Security Account Delegation" in SQL Server Books Online. Also go to Knowledge Base (support.microsoft.com) and see the article "Use Kerberos Authentication for Microsoft SQL Server 2000 Analysis Services." |

**SQL Server Agent Service**

The SQL Server Agent service is used to run the Analysis Services Processing task in a Data Transformation Services (DTS) package, and to perform jobs containing DSO operations executed via Microsoft Visual Basic� Scripting Edition (VBScript) scripts. To ensure that the SQL Server Agent service account has appropriate permissions to perform these tasks, run the SQL Server Agent service under a domain user account (or the NetworkService account in Windows Server 2003) and then add this account to the OLAP Administrators group on the Analysis Services computer.

Detecting permissions problems with SQL Server Agent can be somewhat confusing unless you understand that when you run a SQL Server Agent job interactively (right-click the job in SQL Server Enterprise Manager, and then click **Run**) the security credentials used are not the credentials of the SQL Server Agent service account. When you run a SQL Server Agent job interactively, the security credentials of the user that initiates the job are used. The only time the credentials of the SQL Server Agent service account are used is when the job is actually scheduled (unless you log on using the domain user account used by the SQL Server Agent).

**Migrate the Repository**

The meta data for the objects created in an Analysis Services instance (the cubes, dimensions, and so on) are stored in the Analysis Services repository. By default, this repository is a Microsoft Access database named **msmdrep.mdb** and is stored in the ..\Microsoft Analysis Services\Bin folder on the Analysis Services computer. The Access format is used so that users who do not use SQL Server for relational data can still use Analysis Services. However, if you do use SQL Server, migrating the repository to a SQL Server database adds enterprise-level scalability, support, and security. Migrating the repository also enables you to perform coordinated backups of the repository database with a file-based backup of the Data folder. For more information, see "Backup and Recovery" later in this paper.

Before you migrate the repository, create a dedicated database (such as a database named **OLAPRepository**) using a case-insensitive collation. A dedicated database enables you to back up the repository database on its own schedule. While you can create this dedicated database on a SQL Server instance located on a remote computer, for best performance you should create this database on a local SQL Server instance. To migrate the repository to SQL Server, use the Migrate Repository Wizard and choose Analysis Services native format.

**Important**   Under most circumstances, do not migrate the repository to the **msdb** database, which is the default database selected by the Migrate Repository Wizard. While the **msdb** database is appropriate for a single SQL Server instance dedicated to the Analysis Services repository, it is not appropriate for the typical shared environment. If you select the **msdb** database, the Analysis Services repository is shared with all other SQL Server system-level resources in that instance, such as database maintenance jobs, replication definitions, DTS packages, and execution logs of all different types. By using a dedicated database, you can back up and recover the repository on its own schedule and independent of the other objects stored in the **msdb** database.

After you migrate the repository to a SQL Server database, you cannot migrate it back to a Microsoft Access database. Migrating does not remove the **msmdrep.mdb** database. For added security, you should remove the **msmdrep.mdb** database (by deleting the file using Windows Explorer) after you successfully complete the migration. If the migration fails for any reason, Analysis Services discards any changes and continues using the **msmdrep.mdb** database.

**Tip**: To determine the location of the repository on an unfamiliar Analysis Services instance, right-click the server object in Analysis Manager and then click **Edit Repository Connection String**. This option was added with SP3. Before SP3, you could review the connection string in the registry. However, after SP3, this connection string in the registry is encrypted and using the **Edit Repository Connection String** command in Analysis Manager is the only way to view the current repository.

**Logging and Error Reporting**

Analysis Services records a query log to enable you to analyze query patterns and improve your aggregation design. You can configure the properties of this query log. You can also enable a processing log and enable Analysis Services error reporting.

**Query Log**

To enable the Usage Based Optimization Wizard to design aggregations based on past usage patterns and to enable the Usage Analysis Wizard to generate reports analyzing query usage, Analysis Services records the levels touched by every *N*th query in a query log, which is stored in a Microsoft Access database. By default, every tenth query is logged. The default location for the query log is C:\Program Files\Microsoft Analysis Services\Bin\msmdqlog.mdb. This file, like any log file, should be secured from unauthorized access.

You can change the logging interval (e.g. every *N*th query), stop all query logging, or clear the query log. Setting the logging frequency too low may adversely affect performance. Increasing the logging frequency above 10 might increase performance, particularly if you are on a system with hundreds of concurrent users generating many queries per second. On such a system, Analysis Services attempts to log many queries very quickly, and Access cannot write this volume of information as quickly as a high-performance database system.

If there appears to be considerable activity to the query log, or for additional stability and recoverability, consider migrating the query log to a dedicated SQL Server database. While there is no built-in migration facility within Analysis Services, it is a fairly straightforward process. Simply export the QueryLog table within the **msmdqlog.mdb** Access database to a SQL Server database, edit the **QueryLogConnectionString** registry setting, and restart Analysis Services. If you do so migrate the query log, remember to change your backup and recovery procedures accordingly.

In addition, you should consider clearing the query log after you run the Usage-Based Optimization Wizard consecutively on all cubes on the server. To modify the query log properties, right-click the Analysis server object in Analysis Manager, click **Properties**, and then click the **ogging** tab.

**Note**   The query log records queries on a server-wide basis, not on a per-cube basis.

**Processing Log File**

Analysis Services does not record a processing log by default; you must enable the recording of a processing log file. The processing log file is a system-wide text file containing all of the processing information you see displayed in the **Process** dialog box in Analysis Manager when you process a partition, dimension, or mining model. Recording a processing log file enables you to:

| � | **Troubleshoot problems.**Suppose someone performed a full process of a non-changing dimension at 2 A.M. and later that morning all cubes in the Analysis Services database were knocked off-line. The processing log provides you with an audit trail to identify the problem and the culprit. |
| --- | --- |
| � | **Perform long-term trend analysis.**Suppose every job that runs at 2 A.M. Saturday is failing because a data source is being backed up. Maintaining an audit trail over several months helps you identify the trend and determine its cause. |
| � | **Analyze processing performance.**The processing log file records the time required for processing each dimension, partition, and mining model. You can analyze how the time required for processing these objects changes over time. For example, as you add more aggregations to a partition, the time required to process that partition increases. If you have a fixed length of time during which you must complete all nightly or weekly processing, the processing log file enables you to determine which objects are taking the longest to process and which ones are requiring more time than they required in the past. |
| � | **Recover from closing the interactive dialog box too fast.**It is easy to close the interactive dialog box presented by Analysis Manager when you process an object. The processing log file enables you to review any errors or messages after you have closed the dialog box. |

To enable a processing log file, right-click the Analysis server object in Analysis Manager, click **Properties**, and then click the **Logging** tab. Because the processing log can become quite big, you should periodically rename the old one and have Analysis Services begin a new one (perhaps once per month or once per quarter, depending upon how often you process objects). Renaming the existing processing log file periodically enables you to keep a running history in files of reasonable size.

While the system-wide processing log is normally used for most applications (such as Analysis Manager), at the DSO level you can configure applications to redirect the output to other locations. As a result, you need to be aware that the processing log might not contain all of the processing requests performed on the system. For example, the processing log generated by the DTS OLAP Processing task is redirected to the DTS log file-and its processing will not be captured in the system-wide processing log. The DTS OLAP Processing task is the only Microsoft-supplied application that redirects its activity into a separate log file.

**Tip**   Choose the same location on every server to ensure the file is easy to locate on each of the Analysis Services computers that you are administering.

**Note**   If you are using the Parallel Processing Utility (from the SQL Server 2000 Resource Kit) to process Analysis Services partitions, you can also specify a log file name on the ProcessPartition.exe command line (such as a datetime stamp) and then save these log files for troubleshooting purposes.

**Error Reporting**

When a fatal error occurs in Analysis Services, you can choose to have Analysis Services automatically send an error report to Microsoft. Microsoft then uses this information to improve Analysis Services, treating all user information as confidential. To enable error reporting, right-click the Analysis server object in Analysis Manager, click **Properties**, and then click the **Error Reporting** tab.

**Note**   This option is only available with SP3.

**Performance Configuration Issues**

From an operational prospective, you can improve Analysis Services performance by keeping partition sizes reasonable, setting partition data slices, defining appropriate aggregations on all partitions, and running the Optimize Schema Wizard in the cube editor. For more information on each of these configuration issues, go to the Technet Web site (<http://www.microsoft.com/technet/>) and see "Microsoft Analysis Services Performance Guide."

**Partition Size**

You can divide a cube into multiple partitions if you are using SQL Server 2000 Enterprise Edition (or SQL Server 7.0 Enterprise Edition) to increase query and processing performance. As a general guideline, you should consider increasing the number of partitions if a partition file exceeds 5 GB or 20 million records. Smaller partitions require less time to query by minimizing the amount of data read on a partition scan. Multiple partitions require less overall processing time because each partition is smaller, and some partitions will not have to be processed if the new data does not affect those partitions. To determine the size of each partition, review the partition files in the Data folder. For detailed information on each type of file in the Data folder, see Appendix K, "Data Folder Structure," later in this paper.

**Data Slice**

When you partition a cube, you should define the data slice for each partition using the Partition Wizard. The Partition Wizard does not require you to set this data slice when you create a partition. As a result, it is possible (even easy) to create a partition without setting the data slice. In fact if you simply click through the Partition Wizard pages taking the default values, you end up with a partition without a data slice being set. You should verify that each partition of each cube in each Analysis Services database has a data slice defined. The only exception to this is if you have a cube with only one partition-in that case the data slice should not be set, because you want all of the cube data to be placed in that one partition. To verify that a data slice is defined, edit the partition in Analysis Manager and then step through the Partition Wizard pages. You can use the sample script in Appendix I, "Sample Script to Determine Whether Data Slices Have Been Set," to determine whether a data slice hs been set on each multi-partition cube in an Analysis Services instance.

Defining the data slice enables Analysis Services to quickly eliminate irrelevant partitions from query processing. The data slice identifies the actual subset of data contained in each partition. Unless Analysis Services knows the range of data contained in each partition, it must query each partition, which negates much of the query performance benefit of partitions. To draw an analogy with SQL Server, creating a partition without a data slice is like creating a partitioned view without the CHECK clause. While you can do it, you force the query optimizer to scan all of the partitions in the view because you haven't given it enough meta data to figure out what partition to access when a query is issued. While the Analysis Service's runtime engine does not use a relational query optimizer (it has its own component that accomplishes a similar operation), it uses the data slice in roughly the same way: as meta data to tell it which partitions to scan if an aggregate cannot be used or is not available.

If you partition a cube by month, and have 36 months worth of data (in 36 partitions), and if you don't specify the data slice, then the runtime engine must scan all 36 partitions to answer a query. If you specify the data slice, it could potentially only have to scan 1/36th the amount data, with an obvious improvement in performance.

Setting a data slice also causes Analysis Services to add a join and a WHERE clause to the SQL statement used for retrieving data from the source database during processing. The WHERE clause limits the data retrieved by the SQL statement to the data that belongs in the data slice. For example, if you say that a partition's data slice is June 2003, then Analysis Services adds a join to the time dimension and adds the WHERE clause:

WHERE <month field> = 'June' AND <year field> = '2003'

or whatever the appropriate member/level names are. If you do not define a data slice and you have multiple partitions, Analysis Services does not restrict the data that is retrieved from the source database. Without the data slice, if you just happen to have July 2003 data in the June partition, Analysis Services does not complain, it just double-counts the July 2003 data (for more information, see "Maintaining Partitions" in SQL Server Books Online). By specifying the data slice, the system can add these JOIN and WHERE clauses that assist in maintaining the integrity of the data.

You can suppress the automatic generation of a WHERE clause for all partitions on the Analysis server by modifying the **DataCompressionSettings** registry setting and add the hex value of 0x00100000 to the existing value for this key. If Analysis Services is loading data for each partition from separate tables in the source database, this may yield some performance benefits. However, you should not disable the automatic generation of the WHERE clause unless you are totally sure that the relational database partitioning is 100% correct when loading data.

**Important**   It is vital to reiterate that the **DataCompressionSettings** registry setting is a server-wide setting. You must be 100% certain of the correctness of the data in *all* partitions of *all* cubes on the server in order to use this setting safely. Without the WHERE clause for your protection, double-counting data (or many-times counting of data) may occur, which could lead to server crashes if inconsistent data is processed. If you disable the generation of the WHERclause, you assume all responsibility for enforcing data integrity between the data source and the data slice.

**Tip**   If you are creating rolling monthly partitions as each month closes, you should ensure that the data slice is set for each new partition after it is created.

**Aggregations**

The most effective technique you can use to improve overall query responsiveness (assuming that the Analysis Services computer has sufficient memory and hard disk resources) is to design effective aggregations of fact data in each cube partition. However, too many aggregations will unnecessarily increase processing time without significantly improving query performance.

When you use multiple partitions to increase query and processing performance, it is possible to deploy a new partition with no aggregations. While deploying partitions with different aggregation designs is a common optimization technique, deploying a partition with no aggregations generally indicates an error in deployment that can cause performance problems. You should verify that a minimum number of aggregations exist in each partition. You can quickly determine whether aggregations are defined on a partition by looking at the combined sizes of the <*partition*>.agg.flex.data and <*partition*>.agg.rigid.data files for each partition. The minimum size should be at least 100 KB with most datasets, although this will obviously vary from very small to very large datasets. If you have less than that amount, it is quite possible that you have either no aggregates designed or too few.

**Tip**   Designing too many aggregations will slow processing and designing too few aggregations will slow querying. Ensure that all partitions have a minimum number of aggregations - perhaps 10%.

**Schema Optimization**

Running the **Optimize Schema** tool on a cube eliminates unnecessary joins between dimension tables and fact tables, if certain conditions are met. By default, when you first create a cube, Analysis Services constructs a SQL query against the fact table that is a "*N*+1"-way join (where *N* is the number of dimensions). If you have 5 dimensions, then you have a 6-way join (between the fact table and the 5 lowest-level dimension tables). From the join, the Analysis Services query extracts the lowest-level key. From that key, Analysis Services begins the aggregation process. Having a 6-way join is typically not a major performance problem in most modern relational database systems. However, if your cube has 15 or 20 dimensions, the resulting multi-table join may suffer significant performance problems. Regardless of the number of dimensions in your cube, the Analysis Services query to the relational database is resolved faster and the data flows into Analysis Services more quickly during processing if you eliminate some of these joins.

Fortunately, there is a common design technique that can greatly help the situation. Many star or snowflake schema designs are constructed in such a way that the foreign key that points from the fact table to the lowest-level dimension table is not some random number, but is also the member key itself. If that is true, then Analysis Services can "optimize away the join" and pull the member key directly from the fact table instead of using a join to the lowest-level dimension table.

However, certain conditions must be met for Analysis Services to eliminate a join between a dimension and the fact table. These conditions are:

| � | The dimension must be a shared dimension. |
| --- | --- |
| � | The dimension must have been processed before you optimize the cube schema. |
| � | The member key column for the lowest level of the dimension must contain the keys that relate the fact table and the dimension table, and this must be the only key necessary to relate the fact table to the dimension table. |
| � | The keys in the member key column for the lowest level of the dimension must be unique. |
| � | The lowest level of the dimension must be represented in the cube; that is, the level's **Disabled** property must be set to **No**. The level can be hidden. |

If these conditions are met with respect to a dimension used in a cube, and the cube's schema is optimized using the **Optimize Schema** command, Analysis Services composes a query that does not contain a join to the dimension table in the database when processing the cube. If these conditions are met for all dimensions in the cube, the Analysis server needs to read only the fact table to process the cube. Processing time reductions often can be substantial when this optimization technique is used.

**Note**Cube schema optimization applies to all partitions of the cube, whether the partitions are processed independently or as a group.

So, as a general rule, after you have designed the schema for a cube, you should run the **Optimize Schema** command. It removes the joins that meet the foregoing conditions. Next, you should determine which dimensions were not eliminated from the join and then determine how to meet the required conditions to eliminate the dimension table from the join. If you have partitioned your cube and specified the data slice, the dimension table used for the data slice cannot be eliminated. This join is set to protect you so that no additional, non-data slice data is included in the partition.

If you do optimize away a dimension, you should be aware that the inner join that you have just eliminated had a side effect that may expose problems with your source data. The inner join to the dimension table eliminates fact table records that do not have matching dimension table records (this is what an inner join will do). This means that when you remove the inner join and start using the fact table member keys, you may start seeing processing errors that you were not getting before. When Analysis Services processes a record in the fact table that does not have a corresponding entry in the appropriate dimension table, Analysis Services generates an error.

**Important** If you recreate a cube, add a dimension to a cube, or remove and then re-add a dimension, you must rerun the **Optimize Schema** command to re-optimize the cube. New dimensions are always added un-optimized.

**Verifying the Appropriate Service Pack or Hot Fix Level**

Ensuring that you are working on an Analysis Services instance that has the latest service pack (or hot fix) can assist you in resolving problems. Similarly, when working on an Analysis Services client computer, you need to ensure that the most recent service pack (or any applicable hot fix) has been applied to that client. Unfortunately there isn't a quick and easy way to determine the service pack level (or hot fix) that has been applied to an Analysis Services instance or to an Analysis Services client computer.

To determine the level of service pack, including any hot fixes, there are four different files that you have to be concerned with to determine the level of your installation:

| � | **Analysis Services engine**To determine the version of Analysis Services, locate the msmdsrv.exe file in the Microsoft Analysis Services\Bin folder and then right-click this file to get the version information from the **Properties** dialog box. If the file version is 8.00.760, you have applied SP3. |
| --- | --- |
| � | **DSO**To determine the version of DSO, locate the msmddo80.dll file in the Program Files\Common Files\Microsoft Shared\DSO folder and then right-click this file to get the version information from the **Properties** dialog box. If the file version is 8.00.0760, you have applied SP3. While the version of the msmdsrv.exe and msmddo80.dll files will be identical after a service pack installation, the installation of hot fixes can result in different values for these two crucial files. |
| � | **Analysis Manager**To determine the version of Analysis Manager on a client computer, right-click the Analysis Servers object in Analysis Manager and then click **About Analysis Services**. If the version is 8.0.760, you have applied SP3. If you click **About Microsoft SQL Server Analysis Services** on the **Help** menu in Analysis Manager, the build number returned is the build number of the Analysis Manager Microsoft Management Console (MMC) snap-in. |
| � | **PivotTable Service**To determine the version of PivotTable Service, locate the msolap80.dll file in the Program Files\Common Files\System\Ole DB folder and then right-click this file to get the version information from the **Properties** dialog box. If the file version is 8.00.760, you have applied SP3. (msolap80.dll) |

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Release Management**

To move an Analysis Services database from the development environment to the QA and production environment, you can choose between three Microsoft supported mechanisms or an unsupported DSO/XML scripting utility distributed by Microsoft.

**Microsoft Supported Mechanisms**

Analysis Services provides functionality to help you archive an Analysis Services database on one instance and then restore that database to another Analysis Services instance. Analysis Services also provides functionality to copy and then paste the meta data for an Analysis Services database to a new database. Finally, you can also back up the Data folder and the repository and restore it on the destination server.

**Note**You will also need to change the data source connection properties and the repository connection string after deploying an Analysis Services database on another server.

**Archiving and Restoring**

You can archive an Analysis Services database using Analysis Manager or by using the msmdarch.exe command directly (you cannot archive a single cube within a database). The archive file for an Analysis Services database consists of one or more .cab files that contain the entire contents of the database folder for the database being archived and meta data for the database and its objects from the Analysis Services repository. Data in remote partitions (a feature that is rarely used) and in writeback tables are not stored in the archive file. You must back up the data in remote partitions using a file-based backup method; back up the writeback tables using SQL Server backup.

Because the maximum size for a .cab file is 2 gigabytes (GB) and a file cannot span .cab files (both of these are restrictions of the .cab file technology and are not related to Analysis Services), the maximum size file within the Data folder that can be archived is 2 GB. You can archive more than 2 GB of data, provided that no single file is larger than 2 GB. The msmdarch.exe command will just create multiple .cab files. Since the partition file used to store the MOLAP fact tables is, by far, the largest data file, the partition size is typically the limiting factor. As a result, if you have any individual partition file that is larger than 2 GB, the Analysis Services database cannot be archived. If you are using the Enterprise Edition of SQL Server 2000, increasing your use of partitioning (in other words: adding more partitions, each one smaller) can reduce the size of each partition file below 2 GB to enable you to archive the entire database. (You can use the script provides in Appendix J, "Sample Script to Determine the Analysis Services Edition," to determine the edition of Analysis Services you are using.)

When you restore an archived database to an Analysis Services instance, using either Analysis Manager or msmdarch.exe directly, the Analysis Services file set and its meta data are returned to their states at the time the archive file was created. If you restore a database that has a remote partition, you must process the remote partition. If you restore a database with a write-enabled cube and its writeback table is not available, the cube must be processed before it can be used.

**Note**   Because archive and restore is copying the data along with the metadata, the process can take a long time.

**Tip**   Regularly validate your backup media by performing a restore to a test server. Besides validating the quality of your backup media, regular testing ensures that your backup and restore procedures work properly. Backups without regular validation are worthless and misleading in that they give you a false sense of security. You should validate your backup media at least monthly or quarterly.

**Copying and Pasting**

You can use Analysis Manager to copy the meta data for an Analysis Services database from one instance of Analysis Servicesto another Analysis Services instance, provided that both instances are registered in Analysis Manager. Because only the meta data is copied to the target server using this method for release management, you will need to process the Analysis Services database on the destination server (after updating the data source properties, if necessary) before users can query the data in the new location.

Because copying and pasting is so easy and quick, and because you are frequently only working with a subset of data in the development environment, copying and pasting is generally the quickest way to deploy an Analysis Services database on a different server. The downside is that it requires all of the dimensions, cubes, and partitions to be processed (which must be incurred anyway if the datasets are different). To determine your preferred approach, you need to compare the time and overhead of fully reprocessing the database with the msmdarch.exe archive and restore time. In most cases, full reprocessing is the fastest method. But you will find that it varies based on the underlying infrastructure (for example, fast network between the Analysis server and the source database) and on other uses of the source database (for example, it might be used by other applications and is thus already 80 percent loaded).

**File-Based Backup and Restore**

If neither of the previous two methods is suitable for your situation, you can also use a file-based backup program to back up the entire Data folder and then restore it to the destination folder. With this method, you must deploy all databases within an Analysis Services instance, rather than a single database. If you use this method, you must also back up the repository and then restore the repository on the destination server. While the repository is technically not required to run the OLAP service, it is required to run Analysis Manager and thus to properly administer the server. The meta data in the repository must match the contents and structure of the Data folder. If you have data in remote partitions (a feature that is rarely used) and in writeback tables, you must first back up the data in remote partitions using a file-based backup method and the writeback tables using SQL Server backup, and then restore them before you bring your database back online.

**DSO/XML Scripting Utility**

You can also deploy Analysis Services objects using DSO/XML, an unsupported utility that can create objects using definitions stored in an XML file. This utility uses DSO to query an Analysis Services instance and store XML definitions of the Analysis Services objects in the instance into an XML file. You can then use DSO/XML to read the definitions of these Analysis Services objects from the XML file and recreate them on another Analysis Services instance. Before deploying these stored definitions, you can edit the XML file with any text editor to modify the definitions of these Analysis Services objects, such as object names, data source names, and connection strings.

Microsoft provides the source code with this utility so that you can embed its functionality directly into your management application. To download or obtain more information on DSO/XML, go to [**http://www.microsoft.com/downloads/details.aspx?FamilyID=8d9e7a70-eef4-44c3-a0c5-deece0f8b4b4&displaylang=en**](http://www.microsoft.com/downloads/details.aspx?FamilyID=8d9e7a70-eef4-44c3-a0c5-deece0f8b4b4&displaylang=en).

**Note**   DSO/XML only deploys object definitions. You must process the objects to load the actual data from the data source.

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Change Management**

Change management is the practice of administering changes with the help of tested methods and technologies to avoid introducing new errors and to minimize the impact, if any, on the service level. When implementing change, you should use Microsoft Visual Basic Scripting Edition (VBScript) with Decision Support Objects (DSO) to minimize the possibility of operator error when making the change. The next best method is the DTS Analysis Services Processing task. If neither of these methods can be used, you can make the change manually using Analysis Manager. Change should be tested in a development environment and then again in the QA environment before it is implemented in the production environment. Each change should be documented in the run book to ensure that information in the run book is kept current.

**Important**   Lack of change management can be a major cause of failure and service outages.

Making changes using scripts and DTS packages enables you to employ source code control, such as Microsoft Visual SourceSafe�, to provide version control. Source code control ensures that you can retrieve an older version of a script or package should the need arise, and facilitates team development.

Many Analysis Services installations have multiple individuals with permission to administer the Analysis Services database. Because any administrator can perform any task within Analysis Services and modify any object, it can be useful to track when changes are made to Analysis Services objects. While Microsoft does not provide a direct method for capturing this information, if you migrate the Analysis Services repository to SQL Server, you can add triggers to the repository database to detect when the meta data for an Analysis Services object changes and capture the value of any object before the change to an audit table. The sample script provided in Appendix E, "Sample Script for Creating Repository Audit Triggers," demonstrates how to create such repository audit triggers.

**Note**   Some client tools cache their own meta data. In this case, if you change the meta data in Analysis Services (such as the uniqueness of a level), you may need to notify the client tool that a change has been made and have it update its cached meta data.

**Tip**   If you need a way to quickly take a cube offline in order to make a change (or perform some type of maintenance), you can use virtual cubes. You can have clients connect to the virtual cube and then drop the virtual cube when you need to suspend access in order to make a change. You can then quickly recreate the virtual cube when you are ready for users to access the cube again.

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Security Administration**

One of the key responsibilities of the Analysis Services administrator is ensuring that the data exposed through Analysis Services is secure. All users, whether they are administrators or end users, must be authenticated by the Microsoft Windows operating system before they can access Analysis Services objects. These users can be authenticated directly or via Microsoft Internet Information Services (IIS).

**Administrator Security**

When Analysis Services is installed, the setup program creates the OLAP Administrators local group on the Analysis Services computer and adds the user account of the person installing Analysis Services to this group. All members of the local Administrators group are automatically members of the OLAP Administrators group, regardless of whether they are explicitly added to the OLAP Administrators group.

The OLAP Administrators group is granted the following rights on the Analysis Services computer:

| � | Full control permission to the **Server Connection Info** registry key at HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\OLAP Server. |
| --- | --- |
| � | Write permissions through the MsOLAPRepository$ hidden share (the ..\Microsoft Analysis Services\Bin folder). The MsOLAPRepository$ hidden share is created during setup. Analysis Services uses the hidden share when reads from or writes to the repository when it is stored in an Access database (this is the default location and store for the repository). If you migrate your repository to SQL Server, or modify the remote connection string to the repository manually to specify a different location for the Access database, this hidden share is not needed and can be removed. |
| � | Full control rights to the Bin and Data folders under the ..\Microsoft Analysis Services directory. This includes full control rights to the repository files, Msmdrep.mdb and Msmdrep.ldb.  With clustering, if the Data folder is on a different computer than the computer on which Analysis Services is running, you must ensure that the members of the OLAP Administrators group on the Analysis Services computer have full control rights to this Data folder. This includes the account under which Analysis Services is running. Generally this is accomplished through the use of a domain group. For more information, first see the Service Pack 3 release notes and then go to Microsoft Knowledge Base (support.microsoft.com) and see "PRB: Cannot Process a Cube After You Install SQL Server 2000 Analysis Services Service Pack 3." |

As a result, members of the OLAP Administrators local group can access Analysis Services administrative functions through Analysis Manager or programmatically with DSO. For more information, go to Knowledge Base (support.microsoft.com) and see the article "INF: Permissions That You Must Have to Administer an OLAP Server."

There is only one level of administrative access to Analysis Services. A member of the OLAP Administrators group has complete administrative access to Analysis Services objects, full read access to all cubes and dimensions, and full write access to all write-enabled cubes and dimensions (regardless of any contrary role definitions). A domain or local user that is not a member of the OLAP Administrators group can perform no administrative tasks and has read or write access to the extent permitted based on dimension-level or cell-level security.

**Note**   Clients performing OLAP analysis by issuing MDX queries through an OLE DB provider do not read the registry on the Analysis Services computer and do not require permission on the MsOLAPRepository$ hidden share.

**End-User Security**

End-user security in Analysis Services is based on Windows user accounts and groups. Before you begin configuring end-user security in Analysis Services, you must first create the user accounts and groups within Active Directory. A frequently asked question is whether Analysis Services supports other kinds of authentication. The answer is Yes and No. Yes, it can support other types using HTTP access and IIS (IIS 6.0 includes some new authentication options). However, all these authentication types must ultimately map to a Windows user account in the general sense: including domain accounts, local accounts, the guest account (if enabled), or the built-in NT AUTHORITY\ANONYMOUS LOGON account. Therefore, no, Analysis Services does not support SQL standard security or any similar technology where the authentication is not based on Windows user accounts.

For authentication, Analysis Services uses Security Support Provider Interface (SSPI) as the interface for application security. When you issue a query to Analysis Services, in the connection string (see "Authentication of Direct Connections" in SQL Server Books Online), you specify one of the following SSPI options:

| � | **SSPI=NTLM**  specifies that the normal Windows authentication protocol be used, and enables Analysis Services to interoperate with Windows NT 4.0. Use this provider only when a client computer is connecting directly to an Analysis server. |
| --- | --- |
| � | **SSPI=KERBEROS** specifies that the Kerberos network authentication protocol be used. Kerberos enables interoperability with other security architectures. More importantly to Analysis Services, it supports a more flexible authentication infrastructure. Kerberos is based on "tickets," which greatly reduces the need for repeated authentication on each network resource. The principal advantage of Kerberos for Analysis Services is that its ticket-based approach supports multi-hop architectures: an end user's credentials being passed from the client machine to a Web server, then forwarded to the Analysis server (a three-machine configuration). For more information on Kerberos, see the resources listed in Appendix B, "Resources." |
| � | **SSPI=NEGOTIATE** specifies that the client and Analysis Services dynamically evaluate which is the best authentication SSPI to use. Currently **NEGOTIATE** supports only NTLM and Kerberos; more SSPIs may be added in the future. This technique allows you to design the most flexible application. **NEGOTIATE** requires all computer operating systems to be Windows 2000 or later.  Other SSPI providers are technically possible, but not tested or supported by Microsoft. However the infrastructure is in-place and exposed for integration if required. |
| � | **SSPI=ANONYMOUS** - This option specifies that PivotTable Service (PTS) handle requests in a special manner. When you specify **ANONYMOUS**, PTS does not send authentication credentials to the Analysis server. Instead it tells the server to use Anonymous access, without actually saying what that means. On the server, the OLAP service uses the built-in NT AUTHORITY\ANONYMOUS LOGON account. This technique is useful when you need to support a three-machine configuration-client, Web server (typically using HTTP access) and Analysis server-but don't need or want the infrastructure that Kerberos requires. In this configuration, rather than controlling access on the Analysis server (since all users are logged on using the Anonymous account), use the authentication setup on the Web server's virtual directory.  When you use Anonymous authentication on a Windows XP or Windows 2003 computer, the built-in account is not included in the Everyone group. As a result, you must specify the Anonymous Logon account explicitly when configuring the access in Analysis Manager. For more information, go to Knowledge Base and see the article "INF: Connect to Analysis Services By Using "SSPI = Anonymous" on Windows XP." |

**Security Roles**

After you have created the appropriate Windows user and group accounts, you create security roles within Analysis Services that contain Windows user and group accounts, and define the access each role has to Analysis Services data. You can use database roles, cube roles, and mining model roles.

| � | A **database role** can be assigned to multiple cubes or mining models in a database. Database roles provide default permissions for cube or mining model roles. By default, a database role specifies only read access and does not limit the dimension members or cube cells visible to end users. You can, however, specify read/write access and limit dimension members that are visible and updatable. |
| --- | --- |
| � | A **cube role** applies to a single cube. Defaults in a cube role are derived from the database role of the same name, but some of these defaults can be overridden in the cube role. In addition to the database role features of specifying read/write access and limiting dimension members that are visible and updatable, a cube role also enables you to specify cell-level security. Cell-level security has less memory overhead than dimension security. |
| � | A **mining role** applies to a single mining model. Defaults in a mining role are derived from the database role of the same name, but some of these defaults can be overridden in the mining role.  **Note**   A domain user or group can be a member of multiple roles within Analysis Services. In this case, the effective rights of the user are the combined access characteristics specified in these roles. |

**Dimension-, Cell-, or Application-Level Security**

When you use dimension-level security to limit the dimension members that are visible or updateable, Analysis Services must create a replica dimension in memory when a user connects which reflects the dimension members that user is permitted to see. For example, suppose you have an Account dimension that, at the highest level, has four regions: NorthAmerica, Europe, Asia, and Other. By creating four roles, you can specify which accounts a user can see by placing each user one of four roles: (a total of 16 role combinations).

| � | A user is not in any role: no access is permitted to the dimension at all. This is actually an interesting case. If a user is allowed access to a cube (based on the user's membership in the roles), the user can see the cube as a valid cube, capable of being queried. However, when dimension security is applied, the allowed set is empty in one or more dimensions. This places Analysis Services in a difficult position because Analysis Services cannot tell the user where access is being denied (because that is a security violation in and of itself). As a result, Analysis Services forcibly disconnects the session with the user - and the user receives the purposely ambiguous error message "The connection to the server is lost." Needless to say, this can be confusing. |
| --- | --- |
| � | A user is in one role (4 combinations). |
| � | A user is in two roles (6 combinations, for example {NorthAmerica, Asia} or {Asia, Other}). |
| � | A user is in three roles (4 combinations, for example {NorthAmerica, Europe, Other} or {Europe, Asia, Other}). |
| � | A user is in all four roles (1 combination) and can see all accounts. |

Creating standard roles that can be reused by many different users enables Analysis Services to reuse these dimension replicas stored in memory. However, each time a user accesses a cube with a different combination of dimension members that the user has access to, Analysis Services creates a new replica in memory (from the example above, up to 15 replicas).

Replicas remain in memory until either the Analysis Services service is restarted or the base dimension is processed (full or incremental). There is no other way to unload replica dimensions from memory.

Cell-level security is an alternative to dimension security. If you use cell-level security, end users can see all the dimension members. In the example above, they can see all of the accounts-however, some of the cells are secured (if the user is not a member of any role that allows access to that cell). Because it does not need extra copies of dimensions (replicas), cell-level security does not have this memory overhead. While cell-level security scales better in terms of memory use, dimension-level security yields better overall performance. Cell-level security expressions are evaluated for every cell in the query and are not cached at all. If the MDX expression can be executed quickly, then performance is good. In the following example, an MDX expression simply compares the current member against a constant (only customers in "Europe"):

IIF(Ancestor(Customers.CurrentMember, Region).Name = "EUROPE", 1, 0)

Clearly if the expression is complex and involves a lot of processing, then cell-level security can perform poorly and consume a lot of client resources. Cell-level security expressions are evaluated for every cell in the query, and are not cached.

For more extensive, one-on-one security requirements, use dynamic security with the **UserName** function in the MDX statement to set security. Dynamic security allows you to give the end user a single role: a role that uses the **UserName** function to determine what members a user is permitted to see on a user name-by-user name basis. However, if every user name has a different set of members, then dynamic security potentially has a huge number of replicas.

Dimension security (both standard role-based security and dynamic security) is always performed on the server and is totally transparent to the client. Cell-level security is always performed on the client machine.

In practice, you should use a combination of these different role techniques: standard role-based dimension security, cell-level security, and dynamic dimension security. Where you have simple requirements, use cell-level security. It has the least memory overhead. However, in many areas cell-level security cannot be used because just the exposure of the dimension members is viewed as a breach of security. For example, knowing that your company has a particular customer in Asia might be considered confidential information, regardless of what the sales (or cells) are to that customer. In such cases, you must use dimension-level security. In this case, you can reduce the overhead of dimension-level security by using the minimum number of fixed roles. When designing roles, attempt to group the members so there is as little overlap of members as possible. Attempt to limit users to just one role. If both of these guidelines are followed (as closely as possible), then the largest total combined size of replicas is twice (2X) the size of the base dimension. Only use dynamic security for the subset of your users who really need member-by-member control based on the user name.

For even more control, you might be able to use application-level security. For example, suppose you are implementing a 3-tier Web-based application. Because all data access goes through the middle-tier application, you have an opportunity to add more extensive business rules than Analysis Services supports directly. You can choose to allow only certain kinds of operations within a certain number of days of the monthly closing date. Or, you can choose to allow only a certain type of data access if the end user also has credentials in some other security systems, such as a form-based authentication database, a Lightweight Directory Access Protocol (LDAP) server, or some other kind of third-party tool.

Normally this kind of application-level security is available only if you are writing the application yourself. However, some third-party OLAP tools also provide their own security system. For example, Panorama's Software's Novaview (see their web site at <http://www.panoramasoftware.com/>) has an entire subsystem that adds additional controls for users that are using its thin-client, Web application srver. This kind of support varies from product to product.

**Domain Structure Issues**

Because a user must be successfully authenticated before connection to Analysis Services, there generally must be a common domain structure between the Analysis Services computer and the client. However, if you do not have a common domain structure, you have several choices for overcoming this limitation:

| � | If you are using Analysis Services 2000 Enterprise Edition, you can configure Analysis Services for HTTP access through IIS. For more information, see "Connecting Using HTTP" in SQL Server Books Online or go to the MSDN library (msdn.microsoft.com) and see the article "Improved Web Connectivity in Microsoft SQL Server 2000 Analysis Services." |
| --- | --- |
| � | You can match the user accounts and passwords between non-trusted domains. While effective, this option can require significant management overhead to keep these accounts synchronized over time as passwords change. |
| � | You can enable the Windows guest account. This is not a recommended approach because you have no means of auditing who is accessing what data, and access is not limited necessarily to Analysis Services data. Instead, Microsoft recommends that you use the SSPI  authentication option **ANONYMOUS**, outlined earlier in this section. |

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Service and Availability Management**

Analysis Services administrators are responsible for ensuring that Analysis Services and its data are available for browsing by end users. The level of availability required (the amount of down time that is tolerable) depends on the business impact of the unavailability of the data in the Analysis Services cubes. The level of desired availability is generally defined in a service level agreement (SLA) and determines the elements that must be employed to ensure the agreed-on level of availability.

To ensure an agreed-on level of availability, you must develop an availability plan. When you develop your availability plan, take a holistic, system-wide approach and consider Analysis Services as only one part of the entire IT infrastructure. Consider the hardware components of the computer, all necessary software on the computer, the Microsoft Active Directory infrastructure that supports the Analysis Services installation, and the required personnel. Also consider transient information, such as user names and passwords, as well as product installation information, such as CD keys, distribution points, and original installation media.

When determining the appropriate Analysis Services components to add to your availability plan, assess the following:

| � | Is continuous, 24-hour-a-day query access to Analysis Services data required? If so, how do you process new data without compromising availability? |
| --- | --- |
| � | If continuous query access is not required, how do you ensure that all required processing can be completed within the nightly processing window? How do you handle situations that require an entire cube to be reprocessed (changes to non-changing dimensions)? |
| � | How is the Analysis Services data protected against failure of one or more components on the local computer? Will a full reprocess be required after recovery, or can a full reprocess be avoided? |
| � | How is the Analysis Services data protected against failure within the enterprise? What will be the effect if only some pieces of the infrastructure cannot be recovered, such as Active Directory? |

Each of these elements must be addressed to determine how to achieve the desired level of availability. Once you have determined the elements of your availability plan, you must test each element of the plan to ensure that the plan works properly and smoothly in the face of both anticipated and unanticipated threats to availability. A well-trained staff that is prepared to handle any contingency is an essential part of any disaster recovery plan.

**Service Continuity**

You must consider how you will detect when Analysis Services stops running, so that you can respond to a service outage before your users detect the problem. If you want to grow your own detection mechanism, you can use one or more of the following three options to determine whether Analysis Services is still running:

| � | Poll the server at predetermined intervals, such as every 60 or 120 seconds. |
| --- | --- |
| � | Gather values from Windows Performance Monitor using standard APIs. |
| � | Create a SQL Server Agent job that runs an OpenQuery function against PivotTable Service. The sample scripts provided in Appendixes F and G of this paper demonstrate how to create an Analysis Services linked server and then query it to verify its availability. |

You can also purchase a commercial product that performs this function, such as AppManager for Analysis Services from NetIQ (<http://www.netiq.com/>) or ELM Enterprise Manager from TNT Software (<http://www.tntsoftware.com/>).

**Service Management**

As the next step in your availability plan, consider how to define availability in the service level agreement (SLA). For example, Analysis Services can be unavailable for querying because dimensions are being processed due to reorganization. Should this be considered a service outage? The service issues unique to Analysis Services include the following:

| � | **Outages for dimension maintenance**   If the dimensions in your cubes contain non-changing dimensions, then realignment of customers, product lines, or sales will cause downtime during reprocessing. |
| --- | --- |
| � | **Nightly processing windows**   In nightly processing, query response times will be reduced during incremental processing of partitions. In some cases, a cube partition will be completely unavailable during full processing of that partition. |
| � | **Usage-based optimization**   Running the Usage-Based Optimization Wizard on a regular basis, to add new aggregations based on changing query patterns, can increase the total number of aggregations, which in turn would increase processing times and might ultimately exceed the length of the nightly processing window. |

How should the SLA handle cube unavailability caused by these types of user changes? At what point does unavailability caused by these issues require that you consider a 24-hour-a-day solution? For more information on continuous solutions, see "Implementing a Continuous Analysis Services Solution" later in this paper.

**Backup and Recovery**

Regardless of the other components of your availability plan, regular backups are an essential component. You must also ensure that these backups can be quickly restored if they are needed.

Before you back up your Analysis Services data, you must ensure that Analysis Services is not processing any dimensions, partitions, or mining models. Because Analysis Services performs some processing tasks as background processes, determining when all processing has been completed can sometimes be difficult. Also, you must ensure that another administrator is not changing any of the meta data while you are performing a backup. One way to ensure Analysis Services is quiescent is to stop Analysis Services before you perform the backup. You can use the sample script provided in Appendix H, "Sample Script to Determine When Lazy Processing is Complete," to assist you in determining when all background processing is complete.

**Backup Options**

Analysis Services provides two techniques for backing up an Analysis Services database: archiving and copying files.

**Archiving**

You can archive an Analysis Services database and the repository to one or more .cab file using the **msmdarch** command (msmdarch.exe), either from within Analysis Manager or from a command prompt. **Msmdarch** uses .cab storage algorithms, which limits the size of any single .cab file to 2 GB. As a result, no individual file in the Data folder (such as any single partition) can exceed 2 GB, or else **msmdarch** cannot be used for backup. When using **msmdarch**, always specify a log file location to capture any messages generated during the archive process. If the archive process fails, these messages can help you determine why the archive process failed. However, **msmdarch** does not back up the query log. To back up the query log, perform a file-based backup of the MSMDQLOG.mdb file. If you do not, a new query log is  created from scratch when you start a restored instance.

**Copying Files**

If you cannot use **msmdarch**, you can use a file copy program, such as Windows Backup, to back up all the files in the Data folder. With a file copy backup, you back up all databases on the server. With **msmdarch**, you can back up a single Analysis Services database. In addition, the file copy technique does not back up the repository or the query log file. If you use the file copy technique, you must back up the repository at the same time you back up the Data folder, to ensure that the repository and the data in the Data folder remain synchronized. You must also back up the query log (the MSMDQLOG.mdb file), or begin capturing query information from scratch.

While it is possible to back up just an individual database (by copying the contents of its Data subfolder and the .DBO file), you cannot back up individual portions of the repository. The repository is needed on a full restore, or if the meta data has changed since the last backup. Thus Microsoft recommends that if you are using the file copy technique, you back up the entire Data folder: all databases at the same time.

If you have a Storage Area Network (SAN), you can perform the following steps to create an offline image of the Data folder, the repository, and the query log that can be backed up while maximizing availability of the data for querying.

| 1. | Create a mirror set and then wait until it is fully synchronized. |
| --- | --- |
| 2. | Stop Analysis Services. |
| 3. | Break the mirror. |
| 4. | Restart the service. |

You can then mount the mirrored image as a separate drive and do your file-level backups from there without worrying about consistency.

**Note**The term EMC Corporation uses for this offline image is a Business Continuance Volume (BCV).

**Recovery Options**

Restoration is an all-or-nothing process and is inherently a high-risk activity. This means that you and your staff must thoroughly test and prepare for recovery in a test or QA environment before you have to perform such a recovery in the middle of a crisis. Inadequate training and preparedness can convert an existing problem into a more complex or longer-running problem.

| � | If you back up an Analysis Services database using the **msmdarch** command, you also use **msmdarch** to restore the database. This is the preferred recovery option because **msmdarch** automatically integrates the repository updates. |
| --- | --- |
| � | If you use the file copy technique, you must stop Analysis Services to perform the restoration, and then you must restore the repository database, before you start Analysis Services. If you do not, and the repository contains meta data that is inconsistent with the restored data, you will receive errors when managing the database objects, and you must then reprocess all the cubes in the database. In addition, you must replace the query log file with the backed up version. |

With both backup methods, you must replace the query log file with the backed-up version.

**Note**   Whether you restore using the **msmdarch** command or the file copy technique, you may have issues with security mappings if user names assigned to roles have changed since the backup.

**Implementing a Continuous Analysis Services Solution**

If your availability plan requires that users be able to query the cubes and dimensions on a continuous, 24-hour-a-day basis, there are a number of challenges that you must overcome. These include the following:

| � | **The repository**   If you use multiple servers to assure availability, you must ensure that the repository on each server remains synchronized with the Data folder, which you are also synchronizing. While the repository is not required for query processing, it is required when any structural change is made to the Analysis Services cubes and dimensions. If users are querying a copy of the Data folder on a secondary Analysis Services instance, you must make the change in the original Analysis Services instance and then update the secondary instance (using the file copy technique or **msmdarch**). |
| --- | --- |
| � | **Writeback**   If you enable your cubes or dimensions for writeback, they can only write back to a single location (such as a SQL Server table). This creates a single point of failure (and possibly a performance bottleneck). |
| � | **Processing**   Dimension processing might force cubes offline when structural changes have been made to non-changing dimensions. |

When implementing a continuous Analysis Services solution, Microsoft offers two technologies to help you achieve this goal: Microsoft Cluster Services and Network Load Balancing. As you will see, each offers a different type of availability for your Analysis Services computer.

**Microsoft Cluster Services**

Microsoft Cluster Services (MSCS) enables you to protect your Analysis Services installation against hardware and software failure. With MSCS, you install Analysis Services on two nodes in a cluster with a shared file storage subsystem (with some version of RAID) that contains the Data folder, the repository, and the query log. Only one node in the cluster is active at any point in time. If the active node fails, MSCS fails over to the passive node and starts Analysis Services on that node. Analysis Services on the failover node uses the Data folder, the repository, and query log located on the shared file storage system. This clustering solution ensures that Analysis Services is available for querying in case the primary server in the cluster fails for some reason. However, MSCS does not provide a continuous query solution. You still face the issue of downtime due to processing. In addition, MSCS does not provide any load balancing: the resources in the passive node are only utilized during failover.

MSCS requires specialized hardware, but is a well-proven technology that is familiar to the operations staff of most data centers. For more information, go to Knowledge Base (support.microsoft.com) and see the article "HOW TO: Cluster SQL Server 2000 Analysis Services in Windows 2000."

**Network Load Balancing**

Windows 2000 Network Load Balancing (NLB) enables you to protect your Analysis Services installation against hardware and software failure. It also provides load balancing of queries across multiple servers, and a continuous query availability solution. With NLB, you install multiple instances of Analysis Services on separate computers and then ensure that each has an identical copy of the Data folder and the repository. Each also has a separate query log. The NLB service automatically load-balances query requests among the instances of Analysis Services in the NLB cluster. If any one of these servers fails, NLB simply detects the failure and routes user queries to a server that is running. To support more users with faster response times, simply add more servers to spread the load (and increase availability).

To achieve continuous query capability with NLB, you perform the following steps:

| 1. | Create your server cluster of Analysis Services computers and synchronize the Data folder and the repository on each server. |
| --- | --- |
| 2. | When you need to perform processing, you remove one of the servers in the NLB cluster from NLB cluster and perform the processing on that server. This leaves one or more servers available for querying. |
| 3. | When processing is complete, add the server with the newly processed data to the cluster and remove the other server or servers. |
| 4. | Synchronize the Data folder (and the repository, if any meta data changes were made) on each of these servers with the Data folder (and repository) on the server that was just processed. |
| 5. | Rejoin these servers to the cluster. At all times, a server is available for querying. |

NLB does not require specialized hardware, but does require network expertise (such as knowledge of TCP/IP configuration issues) to configure the NLB cluster. For more information, go to the Microsoft Technet Web site (<http://www.microsoft.com/technet/>) and see "Creating Large-Scale, Highly Available OLAP Sites: A Step-by-Step Guide."

**Disaster Recovery Issues**

Part of your availability solution must be a disaster recovery plan. This plan must provide all the information needed by the staff on duty when the disaster strikes, to enable them to handle every contingency that can be anticipated. You need to address how the operations staff should respond to each type of disaster, the steps that should be taken, andcontact information for the support staff responsible for other portions of the IT system. For example, you must address the following:

| � | In what situations must the dimensions be reprocessed and aggregations recalculated in order to restore the Analysis Services database? |
| --- | --- |
| � | In what situations must the data from the underlying relational tables be reloaded (refreshed) to restore the Analysis Services database? |
| � | What should be done if only a portion of the data in the cubes and dimensions can be recovered? |
| � | What should be done if some other piece of IT infrastructure is not available (such as Active Directory or an IIS server)? |

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Capacity Management**

Analysis Services administrators must understand how Analysis Services is using the memory, disk, processor, and network resources. Its use of these resources may change over time, which means that you should keep a record of your system's use of these resources over time to anticipate the need for additional resources or reconfiguring of existing resources.

**Memory**

Analysis Services uses virtual memory address space as needed, relying on the Windows operating system to map these virtual memory addresses to physical memory. Analysis Services uses memory at startup to load all MOLAP dimension members into memory. Thereafter, usage of all remaining memory fluctuates based on the volume of queries, the number of replica dimensions in memory, and the amount of memory required for processing. Over time, memory use may increase due to dimension growth, new databases that reuse shared dimensions, and the increased number of replica dimensions stored in memory.

This section describes memory capacity issues. For a discussion of memory configuration, see "Memory Settings" earlier in this paper.

**Memory Consumption By Dimensions**

On startup, Analysis Services loads into memory all MOLAP dimension members for all databases on the Analysis server, along with all of their member properties, to help increase query responsiveness. By default, MOLAP, HOLAP, and ROLAP cubes contain MOLAP dimensions. If you create a ROLAP cube and specify ROLAP dimensions, these dimensions are not loaded into memory. New dimensions and dimension members are added to memory as they are created. The amount of memory consumed by an existing dimension in memory is adjusted only during dimension processing. This means that large MOLAP dimensions can consume a significant amount of virtual memory on the Analysis server, reducing the address space that is left for other tasks. The amount of memory required by dimension memory tends to increase over time as new dimensions and dimension members are added to the cubes in the Analysis Services instance.

For an estimate of the memory space required to hold each dimension, you can look at the sizes of the files that hold the dimension structure in the file system. For shared dimensions, this dimension structure information is stored in four types of files: .dim, .dimcr, .dimprop and .dimtree. You can find these files in the database folder for the cube, which is stored in the Analysis Services Data folder. The amount of memory required for dimension memory is approximately equal to the sum of the sizes of these files. For more information on these file types, see Appendix K: "Data Folder Structure."

If you need to estimate the space required before the dimensions have been defined, such as when planning a hardware purchase, you can use the following formula as an approximation:

DimSize = CMembers\*(61 + 4\*CLevels + Size(name) \_

+ Size(key)) + 4\*CProps + Size(props)

where:

**CMembers**

The total number of members in the dimension.

**CLevels**

The number of levels in the dimension, including the All level.

**Size**(*name*)

The average size required to hold the member names. For example, a 10-character string stored as Unicode requires 20 bytes.

**Size**(*key*)

The size required to hold the member key. Example: an integer key requires 4 bytes. If the member name is the same as the member key, **Size**(key) is zero.

**CProps**

The number of member property settings in the dimension for all levels. For example, if a level with 1000 members has two properties on each member, there are 2000 property settings for that level. The member property settings are used to identify which member property values are referred to by the member.

**Size**(*props*)

The size required to hold the distinct member property values for all member levels. Remember that member properties are stored as Unicode strings, and each unique string is only stored once. For example, a customer gender property with possible values of **Male**, **Female**, and **Unknown** requires only 34 bytes of storage (17 characters x 2 for Unicode) no matter how many times they are referred to.

**Note**For the 64-bit version of Analysis Services, the formula above should be 8\*CLevels and 8\*CProps because an integer is 8 bytes, rather than 4 bytes as on a 32-bit system.

Analysis Services uses as much memory as required for dimension memory. You can control the amount of memory used as dimension memory by eliminating unnecessary dimensions, levels, and member properties. In addition, because all dimension information in all cubes on the Analysis server is loaded into memory when Analysis Services starts, you should also eliminate unnecessary test cubes and unused dimensions in any of the databases on the Analysis server to save memory. Analysis Services attempts to prevent large dimensions from using all of the available virtual memory address space by loading very large dimensions in a separate process space with its own virtual memory address space. A very large dimension is one that exceeds the **VLDMThreshold** value in the registry. The default VLDM threshold is 64 megabytes (MB). While using a separate address space for dimensions that exceeds the VLDM threshold does save virtual memory address space for other uses in the main process, overall performance slows when one or more dimensions exceed the VLDM threshold. You should icrease the **VLDMThreshold** value if you have sufficient virtual memory address space. Analysis Services in SQL Server 2000 (64-bit) does not use a VLDM threshold because this version does not have a 3-gigabyte (GB) virtual address limit. For more information on VLDM, see "Very Large Dimension Memory (VLDM) Threshold" earlier in this paper.

**Query Results Cache**

Analysis Services stores data returned by client queries (but not calculated data) in its query results cache. The cleaner thread begins evicting entries from the query results cache when the memory used by the Analysis Services process exceeds the halfway point between the **Memory conservation threshold** and **Minimum allocated memory** settings. As the amount of memory allocated for other Analysis Services uses increases, the amount of memory for the query results cache decreases. If you are running low on memory resources, query response times increase because the query result cache is too small.

**Memory Consumption by Connections**

Analysis Services allocates approximately 32 kilobytes (KB) for each client connection. If Analysis Services is supporting a large number of connections, the memory required for each connection reduces the amount of memory available for the query results cache. When a client specifies that the execution location for a query is the Analysis server (this is called a remote query), additional memory is required to service the query. The amount of memory that can be allocated to a remote query is determined by the value of the **AgentCacheSize** registry key. By default, up to ten percent of memory on the Analysis server can be allocated to each agent cache. Because more than one of these caches can be allocated at the same time (to service multiple clients issuing remote queries), reduce this value when many remote queries are being evaluated to reserve memory for the query results cache. For more information on remote queries, go to the Microsoft Technet Web site (<http://www.microsoft.com/technet/>) and see "Microsoft SQL Server 2000 Analysis Services Performance Guide."

**Memory Consumption By Replica Dimensions**

When a user queries a cube that is secured by dimension security, Analysis Services calculates and loads replica dimensions for each unique combination of security roles actually used by the client. For example, if a user is a member of the two roles of plant users and plant administrators, then the final list of permitted and denied members is the combined list. A replica dimension contains all permitted members plus their siblings, their ascendants, and the siblings of the ascendants; the names and properties of members that the client is not permitted to view are removed.

Replica dimensions are unloaded when the dimension (or cube containing the dimension) is processed, when Analysis Services is restarted, or when role membership changes. Replica dimensions are not unloaded when the client disconnects. Retaining replicas in memory enables replicas to be reused across clients that have the same permissions. Before Analysis Services builds a new replica, it checks to see whether a replica already exists with the same allowed and denied list. If so, the replica is reused. If not, a new replica is created. This design works well unless Analysis Services needs to create many different replicas because there are a large number of security roles (for example, 100) or if dynamic security is used and each user has a different allowed and denied set. In dynamic security, the list of permitted ad denied users is based on a custom MDX statement containing the USERNAME function. Replicas will be shared among users, and with security roles, if the list of allowed and denied member sets is the same.

If a cube contains a large number of security roles, combines roles by user, and has a large number of roles that can be seen by all but a few members, replicas can consume a substantial amount of memory on the Analysis server. In this scenario, use cell-level security rather than dimension-level security to limit the impact of security on memory use and performance.

**Memory Consumption By Shadow Dimensions**

When Analysis Services processes a dimension, it creates the dimension in memory as a shadow dimension until processing is complete. While Analysis Services is processing a dimension that has been processed before, user queries are resolved against the previously existing dimension in memory until the processing transaction commits. After the dimension is processed, the old dimension is released from memory and user queries are resolved against the newly processed dimension. If dimensions are processed as part of cube processing (as a single transaction), the creation of shadow dimensions can have a major impact on memory if the cube has large dimensions. When the dimensions of a cube are processed as a single transaction, a shadow copy of each dimension is stored in memory until the transaction commits. While Analysis Services loads dimensions larger than the VLDM threshold in a separate process address space, the shadow dimensions are always created in the main process address space (which can use a significant amount of memory with large dimensions).

If the Analysis server has insufficient memory, processing all existing dimensions in a single transaction can fail. All dimensions are processed in a single transaction whenever you click **Process all dimensions** or **Process the database** in Analysis Manager, or specifically process all dimensions using Decision Support Objects (DSO). If you are running low on memory, you will have to process the objects requiring processing individually rather than in a single transaction to conserve memory.

**Disk**

The Data folder stores the data for all databases in an Analysis Services instance. The Temporary folder (or folders) stores the temporary files used during processing, if any. The usage of disk space in the Data folder will assist you in determining when partitioning of a cube will be useful. The usage of disk space in the Temporary folder will indicate that memory resources are in short supply.

**Data Folder**

Within the Data folder, Analysis Services creates a separate subfolder for each Analysis Services database. Within each database folder, Analysis Services creates a separate subfolder for each cube within that database. While there are a number of different files created in the subfolder for each cube, there are two file types that you should particularly monitor:

| � | **Partition files**   Each partition file has an extension of fact.data. When a partition file exceeds 5 GB or more than 20 million records, you should begin considering the benefits of dividing the partition up into multiple partitions. While these are two general rules of thumb and may vary with circumstances, clearly smaller partitions can be processed faster than larger partitions. Also, with partitions, you frequently do not have to process all partitions in the cube in response to data change. In addition, smaller partitions can be queried quicker because, if the data slice is set properly, Analysis Services needs to scan less data to resolve many queries. |
| --- | --- |
| � | **Aggregation Files**   The files containing rigid aggregations have an extension of agg.rigid.data. The files containing flexible aggregations have an extension of agg.flex.data. As these files get larger, the time required to process aggregations becomes longer. If you monitor the size of these files over time, you can see trends as they develop. |

**Temporary Folder**

Temporary folders are used as temporary storage for aggregations during processing when the amount of memory available for the process buffer is insufficient to hold aggregations in memory during processing. You should avoid the use of Temporary folders if possible, to maximize performance during processing. However, if Analysis Services has insufficient memory in the process buffer to process a partition, it automatically uses the Temporary folder. As a result, you should monitor Temporary folder usage to detect when memory is insufficient. To monitor temporary file use, set Performance Monitor to collect the **Temp file bytes written/sec** counter for the **Analysis Services:Proc Aggs** object and record it to a log file.

If you are processing partitions in parallel, you may need to process fewer partitions in parallel or process partitions serially to conserve memory and avoid the use of temporary files in order to increase overall performance. If you are processing partitions in parallel, you may run out of virtual address space for these simultaneous operations. If so, you receive out-of-memory errors. If you have insufficient physical memory to back allocated virtual memory, memory paging occurs.

**Processor**

When analyzing processor capacity issues, you must look at both querying and processing.

**Querying**

Analysis Services automatically parallelizes queries submitted by users, to maximize the use of all available processors. Complex queries take more processor resources than simple queries, and queries that Analysis Services resolves at the fact level require more processor resources than queries that Analysis Services resolves using existing aggregations. While it is fairly easy to understand how simple queries benefit from aggregations, sometimes what appears to be a simple query is actually very resource-intensive. Some MDX functions such as TOPCOUNT(), AGGREGATE() and MEDIAN() must touch a large number of cells to resolve, but only return a few values (these are called *wide queries*). For example, a query that returns the top three customers in sales for the current year or the median volume of sales per customer for the current year may require a long time for Analysis Services to resolve in a large cube. Depending on the dimension levels involved in the query and the aggregation design, aggregations may help. However, with queries of this type, it is not obvious from the query results that are returned why Analysis Services took a significant amount of time to resolve these types of queries.

When query performance suffers, you must determine the bottleneck:

| � | Insufficient memory, disk, or processor resources |
| --- | --- |
| � | Improper or insufficient aggregations |
| � | Simple versus complex queries |
| � | Well-written versus poorly written MDX queries |

**Processing**

During processing, Analysis Services parallelizes the processing of each partition, but, by default, Analysis Services processes each partition in a cube serially. Parallel processing results in dramatic performance benefits with large cubes during the initial load of the data warehouse, during full cube processing, and during cube refreshes, provided that the Analysis server has sufficient memory to process several partitions in parallel without using temporary files (and the processing buffer is configured to use the appropriate amount of memory). If there is insufficient memory on the Analysis Services computer to store the aggregations for each partition as they are being processed, Analysis Services uses temporary files to supplement the available memory, which negates the performance benefit you are trying to achieve through the use of parallel processing.

Flexible aggregations (lazy aggregations) are recalculated on a low-priority background thread, which can utilize a significant amount of processor resources until these flexible aggregations are recalculated. The thread runs at a low priority so that the lazy aggregator does not affect query response time. However, because these aggregations are calculated on a background thread, lazy processing may never complete if another request is triggered before the previous one completes, or it might take a very long time if the server is overloaded by queries or other non-Analysis Services processing. The recalculation of flexible aggregations is typically triggered by an incremental update to a changing dimension. When attempting to isolate the use of processor and memory resources on an Analysis Services computer when you are not actively processing an object, do not overlook resources used by this background process.

**Network**

Insufficient network bandwidth can affect both query responsiveness and processing performance. Clients that are connected to Analysis Services using slow connections experience slower query response times from Analysis Services than clients using fast connections. If the relational database is located on a different server from the Analysis server, processing performance can be negatively affected if the network is congested.

For information on optimizing query responsiveness over slow network connections or on the impact of architecture on processing performance, see "Microsoft SQL Server 2000 Analysis Services Performance Guide" on the Technet Web site (<http://www.microsoft.com/technet/>). For more information on network issues that relate to SQL Server, see Chapter 11, "Configuring Microsoft SQL Server on the Network," in *Microsoft SQL Server 2000 Administrator's Companion*. Also see "Monitoring Network Activity" in Microsoft System Monitor online Help.

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Problem and Incident Management**

Problem and incident management with Analysis Services is similar to problem and incident management with other server applications or the Windows operating system infrastructure itself. The Analysis Services administrator must employ best practices to detect problems before they create serious problems and then resolve the problems when they occur. The Analysis Services process log file (which should always be enabled), the DTS error and execution logs, and the Windows application log should be reviewed on a regular basis to attempt to detect potential problems before they become bigger problems. These same logs should be reviewed after incidents to attempt to associate events with incidents and identify patterns that lead up to the failure. The resolution of problems should be documented to help resolve future incidents and also used to train personnel in troubleshooting and understanding symptoms.

Analysis Services presents a number of unique problem and incident management issues:

| � | Detecting who is currently using Analysis Services |
| --- | --- |
| � | Detecting network bandwidth issues |
| � | Identifying connectivity problems and clustering issues |
| � | Understanding common performance problems |

**Problem-Solving Tips**

The following is a list of problems you might encounter when operating an Analysis Services instance, along with possible solutions:

| Symptom | Problem | Solution |
| --- | --- | --- |
| Server cycling at startup. | One or more corrupt dimensions are being loaded into memory. | Review the Windows application log to locate entries with Event ID 129 or Event ID 130. These identify the dimension and database involved. Drop and re-add the corrupt dimensions. |
| Cleaner thread reducing allocated memory when sufficient memory is available. | The Analysis Services memory settings were not increased when additional memory was added. | Increase the **Memory Conservation Threshold** and **Minimum Allocated Memory** settings in Analysis Manager to account for the new memory; these settings are not adjusted automatically. |
| Receiving the message: "Unable to connect to the registry on the server (*SERVERX*), or you are not a member of the OLAP Administrators group" when trying to connect to Analysis Services from Analysis Manager. | | 1. | You are trying to administer an Analysis Services instance with SP3 installed from a version of Analysis Manager that does not have SP3 installed. | | --- | --- | | 2. | The remote registry service is not running on the client computer. | | 3. | You have insufficient permissions to the files in the Data folder. | | 4. | There is a high degree of concurrent administrative activity or processing of partitions in parallel. | | | 1. | Install SP3 on the client computer. | | --- | --- | | 2. | Start the remote registry service on the client computer. | | 3. | Add your user account to the OLAP Administrators group or directly to the Data folder. | | 4. | Install hotfix - for more information, see "FIX: Error Messages May Occur During Parallel Processing of Partitions After You Apply SQL Server 2000Analysis Services SP3" in the Microsoft Knowledge Base. | |
| You see empty or zero-value property entries in the Analysis Services **Properties** dialog box when you view an instance. | The remote registry service is not running on the client computer. | Start the remote registry service on the client computer. |
| Receiving a message such as: "A member with key '20030113' was found in the fact table but was not found in the level 'Day' of the dimension 'ReceivedDate'.; Time:724/2003 10:06:11 AM". | You are missing a member with the key 20030113 in your Received Date dimension. | Add the missing member to your Received Date dimension and reprocess. |
| Receiving the message: "Unable to connect: unspecified error" after installing Microsoft Office XP. | There is a DLL conflict problem. | Use the Office XP repair feature. |
| Unable to see all available cubes (can see some of the Food Mart cubes but not your application databases). | If using the SQL Server 7.0 version of PTS, Analysis Services exposes only those cubes that are not using new SQL Server 2000 features. | Upgrade PTS on the client to SQL Server 8.0 and then install SP3. |

When processing partitions, you must ensure that a dimension is completely processed prior to processing any partitions based on it. You can get into trouble if you process a partition while one of its dimensions is also being processed. The parallel processing utility on the SQL Server 2000 Resource Kit avoids this problem by processing in two passes. The first pass is for dimensions (in the order specified). The utility then makes a second scan for partitions in the same workload. If you process dimensions after or while processing partitions, your cube may be left in state where it cannot be queried.

You must also ensure that all background work in the relational database is completely done prior to processing a dimension based on it. You can get into trouble if the underlying dimension table changes while you are trying to perform a dimension update. This restriction must be enforced by the underlying extract, transform, and load process (the ETL process) that updates your dimension tables.

Finally, an incremental process of a changing dimension kicks off the lazy aggregator background processor. If you incrementally process several changing dimensions at once, it can have serious side effects on the responsiveness of your system for a considerable period of time. For more information, see "Microsoft SQL Server 2000 Analysis Services Performance Guide" on the Technet Web site (<http://www.microsoft.com/technet/>).

**Monitoring User Access**

The only way to monitor queries being executed on a per-user basis is to set the query log frequency to 1 in the **Server Properties** dialog box in Analysis Manager, and then review the contents of the query log to determine the levels touched by the queries in question. While this is easy to configure, it won't always provide an accurate measure of user queries. In particular:

| � | If the same query was already issued on the session, then the query might be answered using the client-side cache. If so, it is not recorded in the query log. The query log was designed in conjunction with the Usage-Based Optimization (UBO) Wizard and not as an auditing tool. The query log records only server-side requests. |
| --- | --- |
| � | If a query is being asked in the context of a series of pyramid queries, then the query may not be recorded in the query log. For example, if a user first performs a detailed query, and then drills up, the second higher-level query may be answerable from the aggregates returned by the first query. In this situation, the second query is not recorded in the query log. |
| � | If you look at the query log table, you will notice that this table records aggregation requests rather than actual MDX statements. To capture the MDX statements that generate the aggregation requests, you must specify the LOGFILE= clause in the PivotTable Service (PTS) connection string. This clause tells PTS (on the client) to record the MDX query in a log file on the client. PTS on the client actually parses and resolves the MDX query into a series of aggregate requests for the Analysis server. While you can request that PTS attempt to remote the execution of the query to the Analysis server (using the ISOLATION LEVEL and EXECUTION LOCATION clauses on the connection string), that does not actually transfer the MDX either. The query is still parsed and resolved into an execution plan on the client-but the execution is actually performed on the Analysis server. In either case, by the time the query gets to the Analysis server, the actual MDX statement is no longer available. The actual MDX statement is only visible on the client-side code inside PTS. |

Besides monitoring query activity, you might want to determine when users connect and disconnect from your server. To log connect and disconnect events in the Windows application log, edit the **AuditEvents** key in the registry (\HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\OLAP Server\CurrentVersion) and change the default value from 0xd (13) to 0xf (15).

Actual user connections (the equivalent to the SQL Server **sp\_who** stored procedure) can be monitored using a low-level TCP/IP monitoring tool, such as the TCPView utility from Sysinternals (<http://www.sysinternals.com/>).

This approach works best if there is a direct client-server connection to Analysis Services (Analysis Services listens on port 2725). In this case, the remote computer name is listed on the connection itself. Also, depending on the tool used, the utility can also disconnect an existing session, emulating a break in the network between the client and the server. This forces Analysis Services to clean up the connection and release any resources taken by it.

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix A: Checklist for Operations**

The following is a consolidated list of all tips and tricks listed in this white paper, along with a page number for where they are discussed more thoroughly.

| Item | Page |
| --- | --- |
| Create a run book for each server. | 3 |
| Document all Analysis Services objects. | 3 |
| Check that Analysis Services has not been installed on a domain controller. | 3 |
| If you need to configure process affinity, use Windows System Resource Manager (WSRM) in Windows Server 2003. | 4 |
| Add a second paging file equal to the amount of physical memory in your computer. | 4 |
| Use the 64-bit versions of Analysis Services and Windows Server 2003 to enable Analysis Services to address more than 3 gigabytes (GB) of memory in the main process space. | 5 |
| Use the **/3 GB** switch with the 32-bit version of Analysis Manager, if supported by the operating system. | 5 |
| Disable unnecessary services, in particular the Indexing service. | 7 |
| Disable virus scanning for the Analysis Services Data and Temporary folders. | 7 |
| If you add memory or enable the **/3 GB** switch in boot.ini, increase the **Memory Conservation Threshold** and **Minimum allocated memory** settings in Analysis Manager. | 9 |
| Disable the VLDM threshold if you have sufficient memory in the main process space. | 10 |
| If you must use VLDM, consider going to the 64-bit version of Analysis Services instead. | 11 |
| If you have sufficient memory, increase the **Process Buffer Size** setting to at least 150 or 200 megabytes (MB) to eliminate the use of temporary files during processing. | 12 |
| Use a RAID array for the Data folder, and allow double the space required for the data, index, and aggregation structures. This allows sufficient space during processing and for refreshing the data. | 13 |
| Use a RAID array for the Temporary folder if temporary files must be used during processing. Alternatively, consider adding a second Temporary folder on a different hard disk. | 13 |
| Configure a logical name for the data source object in Analysis Manager. | 14 |
| Configure a domain user account for the MSSQLServerOLAPService service and add this account to the local OLAP Administrators group. Ensure that this account has sufficient access rights to the data source. | 16 |
| Configure security account impersonation and delegation using Kerberos, if client security credentials must be passed through a middle-tier application. | 17 |
| To use the SQL Server Agent service to automate Analysis Services tasks, add the service account used by the SQL Server Agent service to the local OLAP Administrators group. | 18 |
| Migrate the Analysis Services repository to a dedicated database in SQL Server using a case-insensitive collation. This increases scalability, support, and security. Do not use the default **msdb** database. | 18 |
| Enable a system-wide processing log file to enable troubleshooting and analysis. | 20 |
| Use (or increase the use of) partitions to increase query and processing performance if your partitions exceed 5 GB or more than 20 million records. Partitioning requires that you use SQL Server 2000 Enterprise Edition | 21 |
| Ensure that each partition has a data slice defined for the partition, to increase query performance. | 21 |
| Designing too many aggregations will slow processing; too few aggregations will slow querying. Ensure that all partitions have a minimum number of aggregations - perhaps 10%. | 23 |
| Use the **Optimize Schema** command to eliminate unnecessary joins. | 23 |
| Verify that each computer running Analysis Services and each client computer accessing Analysis Services data or metadata has the latest service pack or appropriate hot fix. | 26 |
| To deploy an Analysis Services database, use msmdarch.exe to archive and then restore the Analysis Services database, provided that no single file is larger than 2 GB. Otherwise copy and paste, use a file-based copy program, or use a third-party utility. | 27 |
| Use scripts and DTS packages where possible to effect change for repeatability and to facilitate the use of source code control. Do not use an interactive tool unless absolutely necessary. | 30 |
| Administrators must be members of the OLAP Administrators group on the Analysis Services computer, and must be able to perform any task within Analysis Services, regardless of any other role restrictions. | 32 |
| If you have many different security roles for end users, use cell-level security rather than dimension- level security, to reserve memory for processing and querying.  For even more control, use application-level security. | 36  37 |
| Use a common (same or trusted) domain structure between clients and Analysis Services. | 37 |
| Determine the level of availability required from your Analysis Services installation, and then determine how to provide that level of availability. | 38 |
| Create a mechanism to detect whn Analysis Services stops running and is no longer available. | 39 |
| Perform regular backups, using either msmdarch.exe or a file-based backup method. Ensure that your backup schedule is complete, ongoing, and regularly validated. | 40, 41 |
| Use a test or QA server to practice restorations to prepare for an emergency. | 42 |
| For continuous availability, consider deploying an NLB cluster rather than an MSCS cluster. | 44 |
| Monitor memory consumption changes over time to detect and respond to memory capacity constraints. | 45 |
| Monitor disk space changes over time, including the use of temporary files, to detect and respond to disk and memory capacity constraints. | 49 |
| Monitor processor usage changes over time to detect querying and processing bottlenecks as they appear. | 50 |
| Use traditional problem and incident management techniques to resolve problems quickly, and then use the information learned to prevent future problems and to train staff. | 52 |

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix B: Resources**

The following books, papers, Web sites, and courses are excellent resources for additional information about operating and maintaining an Analysis Services installation.

**Books**

| � | *MDX Solutions: With Microsoft SQL Server Analysis Services,* George Spofford |
| --- | --- |
| � | *Fast Track to MDX,* Mark Whitehorn, Mosha Pasumansky, Robert Zare |
| � | *Microsoft SQL Server 2000 Resource Kit*, Microsoft Corporation |
| � | *Microsoft SQL Server 2000 Bible*, Microsoft Corporation |
| � | *Microsoft SQL Server 2000 Administrator's Companion*, Microsoft Corporation |

**Papers and Links**

| � | "Microsoft Analysis Services Performance Guide" by Carl Rabeler, Len Wyatt, and Dave Wickert, <http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/ansvcspg.mspx> |
| --- | --- |
| � | "Creating Large-Scale, Highly Available OLAP Sites" by Dave Wickert, <http://www.microsoft.com/downloads/details.aspx?FamilyID=9989a445-142b-4872-ac68-2b50f05228e2&displaylang=en> |
| � | "File: Ptssetup.exe Sample Automatically Downloads and Installs OLAP Client" at  <http://support.microsoft.com/default.aspx?scid=kb;en-us;312876> |
| � | "Microsoft SQL Server 2000 High Availability Series" at <http://www.microsoft.com/technet/prodtechnol/sql/2000/deploy/sqlhalp.mspx> |
| � | "Business Intelligence and Data Warehousing" at <http://www.microsoft.com/sql/evaluation/bi/> |
| � | "INF: How to Enable Analysis Services to Use 3 GB of RAM in the Microsoft Knowledge Base" at <http://support.microsoft.com/default.aspx?scid=kb;en-us;295443> |
| � | "OLAP Scribe" at <http://go.microsoft.com/fwlink/?LinkId=22012> |
| � | "Windows System Resource Manager - Fast Facts" at <http://www.microsoft.com/windowsserver2003/techinfo/overview/wsrmfastfacts.mspx> |
| � | "Glossary of Windows 2000 Services" at <http://www.microsoft.com/windows2000/techinfo/howitworks/management/w2kservices.asp> |
| � | Regmon (product) from Sysinternals, <http://www.sysinternals.com/ntw2k/source/regmon.shtml> |
| � | TCPView (product) from Sysinternals, <http://www.sysinternals.com/ntw2k/source/tcpview.shtml> |
| � | "Microsoft SQL Server 2000 (64-bit) Analysis Services: Why Migrate, and What to Expect If You Do" at <http://www.microsoft.com/technet/prodtechnol/sql/2000/evaluate/ansvcs64.mspx> |
| � | "Use Kerberos Authentication for Microsoft SQL Server 2000 Analysis Services" at <http://support.microsoft.com/default.aspx?scid=kb;en-us;817384> |
| � | "DSO/XML" at [http://www.microsoft.com/downloads/details.aspx?FamilyID =8d9e7a70-eef4-44c3-a0c5-deece0f8b4b4&displaylang=en](http://www.microsoft.com/downloads/details.aspx?FamilyID=8d9e7a70-eef4-44c3-a0c5-deece0f8b4b4&displaylang=en) |
| � | "INF: Permissions That You Must Have to Administer an OLAP Server" at <http://support.microsoft.com/?id=231951> |
| � | "INF: Running OLAP Services on a Domain Controller" at <http://support.microsoft.com/default.aspx?scid=kb;en-us;273836> |
| � | "PRB: Cannot Process a Cube After You Install SQL Server 2000 Analysis Services Service Pack 3" at <http://support.microsoft.com/default.aspx?scid=kb;en-us;812601> |
| � | "Improved Web Connectivity in Microsoft SQL Server 2000 Analysis Services" at <http://msdn.microsoft.com/library/en-us/dnsql2k/html/sql_datapump.asp>. |
| � | "AppManager for Analysis Services" (product) from NetIQ, <http://www.netiq.com/products/am/modules/database/analysisServices.asp> |
| � | "ELM Enterprise Manager" (product) from TNT Software, <http://www.tntsoftware.com/sql> |
| � | "How To: Cluster SQL Server 2000 Analysis Services in Windows 2000" at <http://support.microsoft.com/?id=308023> |
| � | "Registry Entries for SQL Server 2000 Analysis Services" by Tom Mathews, at <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/sql2k_anservregsettings.asp> |
| � | "How to Enable Analysis Server To Use 3 GM of RAM" at <http://support.microsoft.com/default.aspx?scid=kb;en-us;295443> |
| � | "Connection Pooling with SQL Server 2000 Analysis Services" by Dennis Kennedy,  at <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/sql2k_connpooling.asp> |
| � | "FIX Error Messages May Occur During Parallel Processing of Partitions After You Apply SQL Server 2000 Analysis Services SP3" in the Knowledge Base, <http://support.microsoft.com/default.aspx?scid=kb;en-us;814414> |
| � | "HOWTO: Perform a SQL Server 7.0 Distributed Query with OLAP Server" in the Knowledge Base, <http://support.microsoft.com/default.aspx?scid=kb;en-us;218592> |
| � | "INF: Connect to Analysis Services by Using "SSPI=Anonymous" on Windows XP" in the Knowledge Base, <http://support.microsoft.com/default.aspx?scid=kb;en-us;324040> |

**Web Sites**

| � | "Business Intelligence and Data Warehousing" at <http://www.microsoft.com/sql/evaluation/bi/> |
| --- | --- |
| � | "MSDN" at <http://msdn.microsoft.com/> |
| � | "Microsoft SQL Server Accelerator for Business Intelligence" at <http://www.microsoft.com/sql/ssabi/default.asp> |
| � | "Ralph Kimball Associates" at <http://www.ralphkimball.com/html/articles.html> |
| � | "Mosha Pasumansky" at <http://www.mosha.com/msolap/index.htm> |
| � | "SQL Server Magazine" at <http://www.sqlmag.com/> |
| � | "SQL Team" at <http://www.sqlteam.com/> |
| � | "InformIT" at <http://www.informit.com/> |
| � | "SQL Server Central" at <http://www.sqlservercentral.com/> |

**Microsoft Official Curriculum Courses**

| � | "2093: Implementing Business Logic with MDX in Microsoft SQL Server 2000" at <http://www.microsoft.com/traincert/syllabi/2093AFinal.asp>. |
| --- | --- |
| � | "2074: Designing and Implementing OLAP Solutions with Microsoft SQL Server 2000" at <http://www.microsoft.com/traincert/syllabi/2074AFinal.asp> |

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix C: How to Tune the Process Buffer Size**

Perform the following steps to tune the process buffer size on an Analysis server:

| 1. | If you have 4 gigabytes (GB) or more of physical memory on the computer, you are running Microsoft Windows Advanced Server or Windows Datacenter Server, and either large dimensions or large process buffers are causing memory concerns, then enable the **/3GB** switch for the operating system and enable Analysis Services to use this additional memory. |
| --- | --- |
| 2. | Set Performance Monitor to collect the **Temp file bytes written/sec** counter for the **Analysis Services:Proc Aggs** object. |
| 3. | Using Analysis Manager, configure the Analysis server properties to assign the Temporary file folder (on the **General** tab of the **Server Properties** dialog box) to an unused physical drive, and configure the process buffer size (on the **Processing** tab) to a minimal value, such as 32 megabytes (MB). |
| 4. | Restart Analysis Services and then use Performance Monitor or Windows Task Manager to determine what the virtual memory usage stabilizes at for the Analysis Services process (msmdsrv.exe). |
| 5. | Process the cube or partitions under consideration and observe the **Temp file bytes written/sec** counter you added to Performance Monitor. Once the aggregation calculation phase starts, you will start to see I/O to the Temporary files. |
| 6. | Gradually increase the process buffer size and re-process (restarting the Analysis Services service each time) until the **Temp file bytes written/sec** counter shows that the Temporary file is not being used. Then increase the number by 10 percent. If the virtual memory allocation for the Analysis Services service exceeds the **HighMemoryLimit** threshold, increase that value as well. |
| 7. | Repeat these steps for any large partitions (or groups of partitions) to determine the best system-wide process buffer size. |

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix D: Sample Script for Changing the Data Folder Location**

Function changeOLAPDataFolder(strDF)

  Dim objWSHShell

  Dim strKey

  strKey = "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\OLAP

  Server\CurrentVersion\RootDir"

  Set objWSHShell = CreateObject("WScript.Shell")

  objWSHShell.RegWrite( strKey, strDF, "REG\_SZ")

End Function

Function moveRepository(strDestFolder)

'// NOTE: msmdrep.mdf file must already exist in the destination folder

'// for example, C:\Program Files\Microsoft Analysis Services\Data

�//(no ending "\" in folder name)

  Dim strDF

  Dim objWSHShell

  Dim strKey

  ' update the locks directory

  strKey = "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\OLAP

  Server\Server Connection Info\"

  objWSHShell.RegWrite( strKey & "Locks Directory",

  strDestFolder, "REG\_SZ")

  ' update the repository connect string

  strDF = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &

    strDestFolder & "\msmdrep.mdb"

  objWSHShell.RegWrite( strKey & "Repository Connection

  String", strDF, "REG\_SZ")

End Function

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix E: Sample Script for Creating Repository Audit Triggers**

/\*

Analysis Services Repository Audit Triggers

(unsupported, for sample use ONLY)

Dave Wickert, Microsoft

May 19, 2003

Note: This information is provided "AS IS"

with no warranties, and confers no rights.

Warning: Expect to see the following error,

"Cannot add rows to sysdepends for the current stored

procedure because it depends on the missing object

'dbo.LookupHierarchyName'.

The stored procedure will still be created."

This is expected because of the recursive nature of

LookupHierarchyName stored procedure.

You can safely ignore the error message.

\*/

SET QUOTED\_IDENTIFIER OFF

GO

SET ANSI\_NULLS ON

GO

-- Change repository name (as appropriate) in next line

USE AS\_Repository

-- Change above line

GO

---------------------------------------

/\* Audit table \*/

IF EXISTS (SELECT \*, name FROM sysobjects WHERE id = OBJECT\_ID

(N'dbo.OlapObjectsAudit')

AND OBJECTPROPERTY(id, N'IsUserTable') = 1)

DROP TABLE dbo.OlapObjectsAudit

GO

CREATE TABLE dbo.OlapObjectsAudit

  ( Reason VARCHAR (10) NOT NULL ,

    ObjectName NVARCHAR (3000) NOT NULL ,

    ObjectType VARCHAR(50) NOT NULL ,

    ObjectDefinition [ntext] NULL ,

    Updated datetime NOT NULL DEFAULT GETDATE() ,

    ByUser varchar(30) NOT NULL DEFAULT SYSTEM\_USER )

ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

---------------------------------------

/\* LookupHierarchyName function      \*/

---------------------------------------

IF EXISTS (SELECT \*, name FROM sysobjects

WHERE id = OBJECT\_ID(N'dbo.LookupHierarchyName')

AND OBJECTPROPERTY(id, N'IsProcedure') = 1)

DROP PROCEDURE dbo.LookupHierarchyName

GO

CREATE PROCEDURE dbo.LookupHierarchyName

  (  @ID VARCHAR (36) , @FullName NVARCHAR (3000) OUTPUT ) AS

BEGIN

DECLARE @ParentID VARCHAR (36)

DECLARE @ObjectName NVARCHAR (150)

DECLARE @ClassType INT

-- Try and look up ID, first in OlapObjects,

then the inserted temporary table,

then finally the deleted temporary table

SELECT @ParentID = ParentID, @ObjectName = ObjectName,

@ClassType = ClassType FROM OlapObjects WHERE ID = @ID

IF @@ROWCOUNT = 0

BEGIN

    SELECT @ParentID = ParentID,

           @ObjectName = ObjectName,

           @ClassType = ClassType

   FROM #inserted WHERE ID = @ID

    IF @@ROWCOUNT = 0

    BEGIN

SELECT @ParentID = ParentID, @ObjectName = ObjectName,

@ClassType = ClassType FROM #deleted WHERE ID = @ID

IF @@ROWCOUNT = 0

BEGIN

    SET @FullName = N'-parent not found-'

    RETURN -- give up -- return not found marker

END

    END

END

-- cannot detect an arbitrary loop in directed graph,

but can detect a parent=self

IF @ParentID = @ID RETURN (N'-self loop-') �

@ObjectName if so, return object name

SET @FullName = @ObjectName -- return value when

ClassType = 2 (database), which is

always top of the object name hierarchy

IF @ClassType <> 2          -- if not database, then recurse down

BEGIN

    EXEC dbo.LookupHierarchyName @ParentID, @FullName OUTPUT

    SET @FullName = @FullName + ' / ' + @ObjectName

END

END

GO

---------------------------------------

/\* LookupClassType function          \*/

---------------------------------------

IF EXISTS (SELECT \*, name FROM sysobjects WHERE id =

OBJECT\_ID(N'dbo.LookupClassType')

AND OBJECTPROPERTY(id,

N'IsScalarFunction') = 1)

DROP FUNCTION dbo.LookupClassType

GO

CREATE FUNCTION dbo.LookupClassType (@ClassType INT)

RETURNS VARCHAR(50) AS

BEGIN

DECLARE @ObjectType VARCHAR(50)

SET @ObjectType = CASE @ClassType

    WHEN 1  THEN 'Server'

    WHEN 2  THEN 'Database'

    WHEN 3  THEN 'DatabaseRole'

    WHEN 4  THEN 'DatabaseCommand'

    WHEN 5  THEN '-not used-'

    WHEN 6  THEN 'Datasource'

    WHEN 7  THEN 'DatabaseDimension'

    WHEN 8  THEN 'DatabaseLevel'

    WHEN 9  THEN 'Cube'

    WHEN 10 THEN 'CubeMeasure'

    WHEN 11 THEN 'CubeDimension'

    WHEN 12 THEN 'CubeLevel'

    WHEN 13 THEN 'CubeCommand'

    WHEN 14 THEN 'CubeRole'

    WHEN 15 THEN 'VirtualCube'

    WHEN 16 THEN 'VirtualCubeMeasure'

    WHEN 17 THEN 'VirtualCubeDImension'

    WHEN 18 THEN 'VirtualCubeLevel'

    WHEN 19 THEN 'Partition'

    WHEN 20 THEN 'PartitionMeasure'

    WHEN 21 THEN 'PartitionDimension'

    WHEN 22 THEN 'PartitionLevel'

    WHEN 23 THEN 'Aggregation'

    WHEN 24 THEN 'AggregationMeasure'

    WHEN 25 THEN 'AggregationDimension'

    WHEN 26 THEN 'AggregationLevel'

    WHEN 27 THEN 'DatabaseAnalyzer'

    WHEN 28 THEN 'CubeAnalyzer'

    WHEN 29 THEN 'PartitionAnalyzer'

    WHEN 30 THEN 'Collection'

    WHEN 31 THEN 'MemberProperty'

    WHEN 32 THEN 'RoleCommand'

    WHEN 33 THEN 'MiningModel'

    WHEN 34 THEN 'Column'

    WHEN 35 THEN 'MiningModelRole'

    ELSE '-unknown-'

END

RETURN (@ObjectType)

END

GO

---------------------------------------

/\* InsertedRecord function          \*/

---------------------------------------

IF EXISTS (SELECT \*, name FROM sysobjects WHERE id =

OBJECT\_ID(N'dbo.InsertedRecord') AND

OBJECTPROPERTY(id, N'IsInlineFunction') = 1)

    DROP FUNCTION dbo.InsertedRecord

GO

CREATE FUNCTION dbo.InsertedRecord (

    @ID VARCHAR (36) ,

    @Reason VARCHAR (10) ,

    @FullName NVARCHAR (3000) ,

    @ObjectType VARCHAR (50) )

RETURNS TABLE AS

RETURN (SELECT @ID as ID, @Reason as Reason,

@FullName as FullName, @ObjectType as ObjectType)

GO

---------------------------------------

/\* UPDATE trigger                    \*/

---------------------------------------

IF EXISTS (SELECT \*, name FROM sysobjects WHERE

id = OBJECT\_ID(N'dbo.OlapObjects\_Update\_Audit')

AND OBJECTPROPERTY(id, N'IsTrigger') = 1)

    DROP TRIGGER dbo.OlapObjects\_Update\_Audit

GO

CREATE TRIGGER dbo.OlapObjects\_Update\_Audit

ON dbo.OlapObjects

AFTER UPDATE AS

BEGIN

SET NOCOUNT ON

DECLARE @ID VARCHAR (36)

DECLARE @ParentID VARCHAR (36)

DECLARE @ObjectName NVARCHAR (150)

DECLARE @ClassType INT

DECLARE @ObjectType VARCHAR (50)

DECLARE @FullName NVARCHAR (3000)

IF UPDATE(ObjectDefinition)

BEGIN

    SELECT ID, ParentID, ObjectName, ClassType INTO #deleted from deleted

    SELECT ID, ParentID, ObjectName, ClassType INTO #inserted from inserted

    DECLARE inserted\_cursor CURSOR FOR SELECT ID, ParentID, ObjectName,

ClassType FROM inserted

    OPEN inserted\_cursor

    FETCH NEXT FROM inserted\_cursor INTO @ID, @ParentID, @ObjectName,

@ClassType

    WHILE @@FETCH\_STATUS = 0

    BEGIN

-- Get the required data

EXEC dbo.LookupHierarchyName @ID, @FullName OUTPUT

SET @ObjectType = dbo.LookupClassType(@ClassType)

-- Insert the audit record

INSERT INTO dbo.OlapObjectsAdit

(Reason, ObjectName, ObjectType, ObjectDefinition)

SELECT i.Reason, i.FullName, i.ObjectType, olapo.ObjectDefinition

FROM dbo.InsertedRecord

(@ID, 'UPDATE', @FullName, @ObjectType) i

INNER JOIN dbo.OlapObjects olapo ON i.ID = olapo.ID

-- next item in inserted rowset

FETCH NEXT FROM inserted\_cursor INTO @ID, @ParentID,

@ObjectName, @ClassType

    END

    CLOSE inserted\_cursor

    DEALLOCATE inserted\_cursor

   END

END

GO

---------------------------------------

/\* INSERT trigger                    \*/

---------------------------------------

IF EXISTS (SELECT \*, name FROM sysobjects WHERE id =

OBJECT\_ID(N'dbo.OlapObjects\_Insert\_Audit')

AND OBJECTPROPERTY(id, N'IsTrigger') = 1)

    DROP TRIGGER dbo.OlapObjects\_Insert\_Audit

GO

CREATE TRIGGER dbo.OlapObjects\_Insert\_Audit

ON dbo.OlapObjects

AFTER INSERT AS

BEGIN

SET NOCOUNT ON

DECLARE @ID VARCHAR (36)

DECLARE @ParentID VARCHAR (36)

DECLARE @ObjectName NVARCHAR (150)

DECLARE @ClassType INT

DECLARE @ObjectType VARCHAR (50)

DECLARE @FullName NVARCHAR (3000)

SELECT ID, ParentID, ObjectName, ClassType INTO #deleted from deleted

SELECT ID, ParentID, ObjectName, ClassType INTO #inserted from inserted

DECLARE inserted\_cursor CURSOR FOR SELECT ID, ParentID, ObjectName,

ClassType FROM inserted

OPEN inserted\_cursor

FETCH NEXT FROM inserted\_cursor INTO @ID, @ParentID,

@ObjectName, @ClassType

WHILE @@FETCH\_STATUS = 0

BEGIN

    -- Get the required data

    EXEC dbo.LookupHierarchyName @ID, @FullName OUTPUT

    SET @ObjectType = dbo.LookupClassType(@ClassType)

    -- Insert the audit record

    INSERT INTO dbo.OlapObjectsAudit

(Reason, ObjectName, ObjectType, ObjectDefinition)

    SELECT i.Reason, i.FullName, i.ObjectType, olapo.ObjectDefinition

    FROM dbo.InsertedRecord(@ID, 'INSERT', @FullName, @ObjectType)

i INNER JOIN dbo.OlapObjects olapo ON i.ID = olapo.ID

    -- next item in inserted rowset

    FETCH NEXT FROM inserted\_cursor INTO @ID,

@ParentID, @ObjectName, @ClassType

END

CLOSE inserted\_cursor

DEALLOCATE inserted\_cursor

END

GO

---------------------------------------

/\* DELETE trigger                    \*/

---------------------------------------

IF EXISTS (SELECT \*, name FROM sysobjects WHERE id =

OBJECT\_ID(N'dbo.OlapObjects\_Delete\_Audit') AND

OBJECTPROPERTY(id, N'IsTrigger') = 1)

    DROP TRIGGER dbo.OlapObjects\_Delete\_Audit

GO

CREATE TRIGGER dbo.OlapObjects\_Delete\_Audit

ON dbo.OlapObjects

AFTER DELETE AS

BEGIN

SET NOCOUNT ON

DECLARE @ID VARCHAR (36)

DECLARE @ParentID VARCHAR (36)

DECLARE @ObjectName NVARCHAR (150)

DECLARE @ClassType INT

DECLARE @ObjectType VARCHAR (50)

DECLARE @FullName NVARCHAR (3000)

SELECT ID, ParentID, ObjectName, ClassType INTO #deleted from deleted

SELECT ID, ParentID, ObjectName, ClassType INTO #inserted from inserted

DECLARE deleted\_cursor CURSOR FOR SELECT ID, ParentID, ObjectName,

ClassType FROM deleted

OPEN deleted\_cursor

FETCH NEXT FROM deleted\_cursor INTO @ID,

@ParentID, @ObjectName, @ClassType

WHILE @@FETCH\_STATUS = 0

BEGIN

    -- Get the required data

    EXEC dbo.LookupHierarchyName @ID, @FullName OUTPUT

    SET @ObjectType = dbo.LookupClassType(@ClassType)

    -- Insert the audit record

    INSERT INTO dbo.OlapObjectsAudit (Reason, ObjectName, ObjectType)

    VALUES ('DELETE', @FullName, @ObjectType)

    -- next item in inserted rowset

&nbs;   FETCH NEXT FROM deleted\_cursor INTO @ID,

@ParentID, @ObjectName, @ClassType

END

CLOSE deleted\_cursor

DEALLOCATE deleted\_cursor

END

GO

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix F: Sample Script for Creating an OLAP Linked Server**

USE master

GO

/\* Additional examples are available at:                         \*/

/\* http://support.microsoft.com/default.aspx?scid=kb;enus;218592 \*/

/\* --------------------------------------------------------------\*/

/\* Remove any previous references to the linked server \*/

EXEC sp\_dropserver 'LINKED\_OLAP'

EXEC sp\_addlinkedserver

     @server='LINKED\_OLAP',   -- local SQL name given to the linked server

     @srvproduct='',          -- not used

     @provider='MSOLAP.2',    -- OLE DB provider

(the .2 means the SQL2K version)

     @datasrc='localhost',    -- analysis server name (machine name)

     @catalog='Foodmart 2000' -- default catalog/database

select \* from openquery

( LINKED\_OLAP, 'Select {[Measures].[Unit Sales]}

on columns from [Sales] ')

select \* from openquery

( LINKED\_OLAP, ' with member [Measures].[Store Profit Rate]

as ''([Measures].[Store Sales]-[Measures].[Store

Cost])/[Measures].[Store Cost]'', format = ''#.00%''

select {[Measures].[Store Cost],[Measures].[Store

Sales],[Measures].[Store Profit Rate]} on columns,

Order([Product].[Product Department].members,

[Measures].[Store Profit Rate], BDESC) on rows

from Sales where ([Time].[1997]) ')

select \* from sysobjects order by name

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix G: Sample Script To Verify Analysis Services Availability**

USE master

GO

/\* Remove any previous references to the linked server \*/

EXEC sp\_dropserver 'LINKED\_OLAP'

EXEC sp\_addlinkedserver

     @server='LINKED\_OLAP',   -- local SQL name given to the linked server

     @srvproduct='',          -- not used

     @provider='MSOLAP.2',    -- OLE DB provider

(the .2 means the SQL2K version)

     @datasrc='localhost',    -- analysis server name (machine name)

     @catalog='Foodmart 2000' -- default catalog/database

select \* from openquery

( LINKED\_OLAP, 'Select {[Measures].[Unit Sales]} on columns from [Sales] ')

select \* from openquery

( LINKED\_OLAP, ' with member [Measures].[Store Profit Rate]

as ''([Measures].[Store Sales]-[Measures].[Store

Cost])/[Measures].[Store Cost]'', format = ''#.00%''

select {[Measures].[Store Cost],[Measures].[Store

Sales],[Measures].[Store Profit Rate]} on columns,

Order([Product].[Product Department].members,

[Measures].[Stoe Profit Rate], BDESC) on rows from Sales

where ([Time].[1997]) ')

select \* from sysobjects order by name

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix H: Sample Script to Determine When Lazy Processing is Complete**

You can call the following script from a batch file to determine when lazy processing is complete. For example, the batch file could contain the following batch:

CScript LazyProcessing.vbs "Localhost" "Foodmart 2000" "HR"

The batch would call the following .vbs file to determine when lazy processing was complete in the HR cube in the Foodmart 2000 database on the local computer.

'File: LazyProcessing.vbs

Option Explicit

'/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' File: LazyProcessing.vbs

'Desc: This sample script displays the lazy aggregator's

progress for a specified cube (all partitions)

'

' Parameters: None

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Call GetLazyProcessing

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' Helper functions

Function ConvertState(dsoState)

    Const olapStateNeverProcessed = 0

    Const olapStateStructureChanged = 1

    Const olapStateMemberPropertiesChanged = 2

    Const olapStateSourceMappingChanged = 3

    Const olapStateCurrent = 4

    Select Case dsoState

        Case olapStateCurrent

            ConvertState = "Current"

        Case olapStateMemberPropertiesChanged

            ConvertState = "Properties changed"

        Case olapStateNeverProcessed

            ConvertState = "Never processed"

        Case olapStateSourceMappingChanged

            ConvertState = "Source mapping changed"

        Case olapStateStructureChanged

            ConvertState = "Structure changed"

        Case Else

            ConvertState = "Unknown state"

    End Select

End Function

Sub GetLazyProcessing()

    Dim bResult

    Dim strMsg

    If Wscript.Arguments.Count <> 3 Then

Msgbox "Invalid number of arguments. This script must be \_

called with three arguments." & VbCRLF & VbCRLF & \_

               "Usage is: (DOS prompt) CScript \_

LazyProcessing.vbs 'Server' 'Db' 'Cube'" & VbCRLF & \_

               "e.g. CScript LazyProcessing.vbs ""Localhost"" \_

               ""Foodmart 2000"" ""HR"" ", , \_

                   "Invalid LazyProcessing caling arguments"

Exit Sub

    End If

    Dim sServer : sServer = Wscript.Arguments(0)

    Dim sDb     : sDb = Wscript.Arguments(1)

    Dim sCube   : sCube = Wscript.Arguments(2)

    bResult = LazyProcessing(sServer,sDb, sCube, strMsg)

    If bResult Then

Msgbox strMsg, , "Get Lazy Processing information"

    Else

Msgbox "Error-" & strMsg, , "Error - Get Lazy Processing information"

    End If

End Sub

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' The real work . . .

Function LazyProcessing(strAnalysisServer, strOlapDb, strCube, strMsg)

    Dim dsoServer : Set dsoServer = CreateObject("DSO.Server")

    Dim dsoDB, dsoCube, dsoPartition

    LazyProcessing = False ' assume we fail

    (strMsg will contain the error text)

    strMsg = ""

    ' VBScript does not support direct use of enumerated constants.

    ' However, constants can be defined to supplant enumerations.

    Const olapStateCurrent = 4

    ' Connect to the Analysis server.

    On Error Resume Next

    dsoServer.Connect strAnalysisServer

    ' If connection failed, then end the script.

    If Err.Number <> 0 Then

       strMsg = Err.Description

       Err.Clear

       Exit Function

    End if

    On Error Goto 0

    ' Find the database on the server.

    If (dsoServer.mdStores.Find(strOlapDB)) = 0 Then

       strMsg = "Database '" & strOlapDB & "'

       not found on '" & strAnalysisServer & "'."

       Err.Clear

       Exit Function

    End If

    Set dsoDB = dsoServer.mdStores(strOlapDB)

    ' Find the cube.

    If (dsoDB.mdStores.Find(strCube)) = 0 then

       strMsg = "Cube '" & strCube & "'

       not found in database '" & strOlapDB & "'."

       Err.Clear

       Exit Function

    End If

    Set dsoCube = dsoDB.MDStores(strCube)

    ' Validate the state of the cube

    if dsoCube.State <> olapStateCurrent Then

strMsg = "   Cube '" & strCube & "'

         state is: " & ConvertState(dsoCube.State) & VbCRLF

strMsg = strMsg & "     Which cannot be checked." & VbCRLF

  Err.Clear

  Exit Function

    End If

    ' Loop through each partition in the cube

    strMsg = ""

    For Each dsoPartition in dsoCube.Partitions

' Only check if the partition's state is current

' and, then, if lazy processing is ongoing.

' Normally, since the lazy aggregator is single threaded,

' only one partition is being processed at a time,

' but we won't assume that at this point.

If dsoPartition.State = olapStateCurrent Then

    If dsoPartition.LazyOptimizationProgress <> 100 Then

' We are in-progress -- output the % complete

    strMsg = strMsg &"   Partition: " & dsoPartition.Name & " is " \_

& CStr(dsoPartition.LazyOptimizationProgress) & "% complete." & vbCRLF

    End If

End If

    Next

    If Len(strMsg) = 0 Then

strMsg = "Cube: " & dsoCube.Name & " is complete." & vbCRLF

    Else

strMsg = "Cube: " & dsoCube.Name & vbCRLF & strMsg

    End If

    LazyProcessing = True ' we succeeded !

    Set dsoCube = Nothing

    Set dsoDB = Nothing

    dsoServer.CloseServer

    Set dsoServer = Nothing

End Function

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix I: Sample Script to Determine Whether Data Slices Have Been Set**

Option Explicit

'/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

' File: CheckDataSlice.vbs

'

'Desc: This sample script scans through each database, each cube, looking

'      for multi-partition cubes for which partition data slices have

'      not been defined. It doesn't validate that the slice makes any sense

'      or is consistent across all partitions -- just that partition

'      slices are defined for all multi-partition cubes.

'

' Parameters: None

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Call CheckDataSlice

Sub CheckDataSlice()

   'On Error Resume Next

   Dim strResults : strResults = ""

   Dim strAnalysisServer

   Dim dsoServer, dsoDB, dsoCube

   ' Initialize server name - you could modify this script to

' pass the server name as a parameter from the command line

   strAnalysisServer = "LocalHost"

   ' VBScript does not support direct use of enumerated constants.

   ' However, constants can be defined to supplant enumerations.

   Const stateFailed = 2

   Const olapEditionUnlimited = 0

   ' Connect to the Analysis server.

   Set dsoServer = CreateObject("DSO.Server")

   dsoServer.Connect strAnalysisServer

   ' If connection failed, then end the script.

   If dsoServer.State = stateFailed Then

      MsgBox "Error-Not able to connect to '" & strAnalysisServer & " \_

' Analysis server.", ,"CheckDataSlice.vbs"

      Err.Clear

      Exit Sub

   End if

   ' Certain partition management features are available only

   ' in the Enterprise Edition and Developer Edition releases

   ' of Analysis Services.

   If dsoServer.Edition <> olapEditionUnlimited Then

MsgBox "Error-This feature requires Enterprise or \_

Developer Edition of SQL Server to manage partitions." \_

, , "CheckDataSlice.vbs"

      Exit Sub

   End If

   ' Ok -- now do the real work -- accumulating the results for each cube

   For Each dsoDB in dsoServer.mdStores

      For Each dsoCube in dsoDB.mdStores

         strResults = CheckCube (dsoCube, strResults)

  &bsp;   Next

   Next

   If Len(strResults) = 0 Then strResults = "  None found." & VbCrLf

   strResults = "Partitions missing a data slice:  " & VbCrLf & strResults

   strResults = strResults & "On " & strAnalysisServer

   MsgBox strResults, , "CheckDataSlice.vbs"

End Sub

Function CheckCube (dsoCube, strResults)

   CheckCube = strResults

   Const clsCube = 9

   Const sbclsRegular= 0

   If (dsoCube.ClassType <> clsCube) Then Exit Function

' must be a cube

   If (dsoCube.SubClassType <> sbclsRegular) Then Exit Function

' must be regular cube

   If (dsoCube.mdStores.Count < 2) Then Exit Function

' Only continue if this is a multi-partition cube

   Dim dsoPartition

   For Each dsoPartition in dsoCube.mdStores

      If Not AnySlice(dsoPartition) Then

    CheckCube = CheckCube & "      " & dsoPartition.Name & " in Cube \_

            '" & dsoPartition.Parent.Name & "' (Database: " \_

            & dsoPartition.Parent.Parent.Name & ")" & VbCrLf

      End If

   Next

End Function

Function AnySlice (dsoPartition)

   Dim dsoDimension, dsoLevel

   AnySlice = False ' assume we fail

   For Each dsoDimension in dsoPartition.Dimensions

      For Each dsoLevel in dsoDimension.Levels

         If Len(dsoLevel.SliceValue) <> 0 Then

            AnySlice = True ' we found a slice !

            Exit Function

         End If

      Next

   Next

End Function

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix J: Sample Script to Determine the Analysis Services Edition**

The following code example checks the **Edition** property of a **clsServer** object to determine feature support.

'VB syntax:

' Dim dsoServer As New DSO.Server

' Enumerated olapXXXXXXX constants already defined when

'       you added the DSO COM object to your project

'VBS syntax:

Dim dsoServer

Set dsoServer = CreateObject("DSO.Server")

Const olapEditionUnlimited = 0

Const olapEditionPivotOnly = 1

Const olapEditionNoPartitions = 2

Const olapEditionError = &HFFFFFFFF

'-------------------------------------------------------

' Connect to the local Analysis server.

dsoServer.Connect "LocalHost"

' Check the Edition property.

Select Case 0 'dsoServer.Edition

    Case olapEditionUnlimited

        ' Insert code for Enterprise Edition features.

Msgbox "EE"

    Case olapEditionPivotOnly

' Reserved for future use.

        Msgbox "Reserved"

    Case olapEditionNoPartitions

        ' Insert code for Standard Edition features.

Msgbox "Standard Edition"

    Case oapEditionError

        ' An error occurred while retrieving this information.

Msgbox "An error occurred"

    Case Other

' Invalid return value

Msgbox "Invalid return value seen"

End Select

dsoServer.CloseServer

Set dsoServer = Nothing

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Appendix K: Data Folder Structure**

The Analysis Services Data folder stores the multidimensional structures for the objects defined on the Analysis Services computer. Within the Data folder, Analysis Services creates an ODB file and a folder for each database. The ODB file contains runtime database information for the database extracted from the repository when Analysis Services starts. The file extension stands for "object database". The Analysis Services runtime engine (msmdsrv.exe) reads database information from the ODB file while Analysis Services is running, not from the repository.

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Database Folder**

The following table describes the database object files within each database folder.

| Object | File Type | Description |
| --- | --- | --- |
| Data source | SRC | Data source meta data. One file for each data source. |
| Database role | ROLE | Information about the role. One file for each database role. |
| Cube | MDL | Cube meta data. One file for each physical or virtual cube |
| Dimension | DIM | Minimal information about the dimension. One file for each dimension is created during processing. For shared dimensions, the name of the file is the name of the dimension with the appropriate extension for the file type. For private dimensions, each file has thefollowing naming convention:  <*cube*>^<*private dimension name*> |
|  | DIMCR | Dimension custom rollup formulas. One file for each dimension. |
|  | DIMPROP | Member properties. One file for each dimension. |
|  | DIMTREE | The hierarchy structure for the dimension, including the dimension members themselves. One file for each dimension. |

In addition, a folder is created for each cube, physical or virtual, within each database.

**Note**   "Minimal information" means a subset of the available meta data stored in the repository. This is just the information the Analysis Services runtime engine needs to perform processing and respond to queries.

[Top of page](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/anservog.mspx#top)

**Cube Folder**

The following table describes the files created within each cube folder.

| Object | File Type | Description |
| --- | --- | --- |
| Aggregations | AGG.FLEX.DATA | Flexible aggregations for each partition. One file for each partition. |
|  | AGG.RIGID.DATA | Rigid aggregations for each partition. One file for each partition. |
| Partition extents | AGG.FLEX.MAP | Bitmap indexes for flexible aggregates to the fact table, one for each extent created during processing. Each extent contains 65,000 records (without duplicates). Each file is named using the following convention:  <*partition name*>.<*extent#*> |
| nbsp; | AGG.RIGID.MAP | Bitmap indexes for rigid aggregates to the fact table. |
|  | FACT.MAP | Bitmap indexes for dimension members to the fact table. |
| Facts | FACT.DATA | Fact table, which may not map one-for-one with the relational database image of the fact table if the two are at different grains. |
| Partition | PRT | Minimal information about a partition, one file for each partition. |
|  | PDR | Catalog / directory for a partition, one file for each partition. |
| Cube role | SEC | Cube level security settings, one file for each cube role. The Analysis Services runtime engine reads security information from this file. |