A Solution for Collecting Analysis Service Performance Data for Performance Analysis

This document explains how to use the files in this Codeplex project to collect SQL Server 2008 Analysis Services performance data from any or all of the following data sources and store the collected data into a single SQL Server 2008 relational database for performance analysis:

* SQL Server Profiler Analysis Services trace data
* Performance Monitor counter data
* Analysis Services Dynamic Management Views (DMV) data

The files in this Codeplex project will install all of the data collection components on a specified server to monitor a specified instance of Analysis Services instance and collect the performance data to the MDW database on the monitoring server. The installation of these components is discussed in the section entitled: **Installing the Data Collection Solution**. This Codeplex project also includes sample Reporting Services reports utilizing SQL Server stored procedures that correlate and analyze the collected Analysis Services performance data. These sample reports are discussed in the section entitled: **Working with the Sample Reports**. In addition, a Microsoft Excel spreadsheet containing the performance metrics as recorded by the ASCMD command-line utility is also included in this Codeplex project.

**For More Information**: For information about using the collected data to address Analysis Services performance issues, see the SQL Server 2008 Analysis Services Performance Guide: <http://sqlcat.com/msdnmirror/archive/2008/11/14/sql-server-2008-analysis-services-performance-guide-now-available.aspx> .

# Data Collection Solution Overview

This section discusses how each Analysis Services performance data type is collected into the MDW database. The installation steps are detailed in the section entitled: **Installing the Data Collection Solution**.

## SQL Server Profiler Trace Data

This Data Collection solution uses an Analysis Services server-side trace to write trace data to a specified share in the file system and an Integration Services package to periodically load these traces files into the MDW database on a specified SQL Server instance.

### Collecting Analysis Service Trace Data to Trace Files

The **Analysis Services Performance and Error Trace** server-side trace is designed to collect the minimum information required to analyze performance problems related to querying or processing, as well as collecting error information (as total of 5 events are collected). The following events are captured:

* Error
* Progress Report End
* Progress Report Error
* Query End
* Query Subcube

**Note**: The SQL Server Profiler template that was used to define this server-side trace is included in the Codeplex project.

The server-side trace starts whenever Analysis Services starts, writes trace information to the specified location in the file system (to either a local folder or a remote location specified with a UNC path), and creates new files whenever a specified maximum size is reached (100 MB is current default value) or whenever Analysis Services is restarted. We chose to use a server-side trace to minimize the performance impact on the Analysis Services instance and to seamlessly handle server restarts (the server-side trace writes to a new file when Analysis Services is restarted).

**Important**: The times written to the file system by the server-side trace are written in local time. The Integration Services package that loads the trace files into the specified SQL Server database calculates the DATETIMEOFFSET value for each row and records this value in an additional column in the destination table. Calculating this value facilitates the correlation of Profiler trace data with the other performance data collected being collection The Management Data Warehouse data collector records information using the new DATETIMEOFFSET datatype and the Analysis Services DMVs record time using GMT time.

### Loading Trace Files by Using an Integration Services Package

The **Load\_SSAS\_Trace\_Data** Integration Services package is called by the **SSAS\_Trace** SQL Agent job to periodically upload the Analysis Services trace data from the file system into the MDW database in the specified SQL Server instance. By default, this job runs every 10 minutes and when you stop the data collection using the Run\_Stop\_Collection.cmd batch file.

**Important**: The currently active trace is ignored by the Integration Services package. After each file is loaded, it is renamed for archive purposes.

When loading this trace data, the Integration Services package calls a number of stored procedures to do the following:

1. Order all of the data being loaded into the AS\_TraceData table by SessionID and EndTime (including all query data left in the AS\_TraceData table from previous loads related to partially completed queries)
2. Separate processing and querying data into separate tables (data in the trace not related to MDX queries or Analysis Services processing operations is not loaded).
3. Calculate performance information about each query. For each unique execution of a query against a cube, the following information is calculated and recorded:
   1. The amount of formula engine time
   2. The amount of storage engine time
   3. The aggregations used (if any)
   4. The number of partitions queried
   5. The number of cached and non-cached subcubes utilized
   6. The subcubes vector for each query.

**Important**: More detail about each stored procedure, view and table used by this package is available in **Appendix A: Working with the Files**. For more information about understanding the data in the trace files, see [Identifying and Resolving MDX Query Performance Bottlenecks in SQL Server 2005 Analysis Services](http://sqlcat.com/whitepapers/archive/2007/12/16/identifying-and-resolving-mdx-query-performance-bottlenecks-in-sql-server-2005-analysis-services.aspx).

**Note**: To further minimize the performance impact when running this server-side trace, we recommend that you disable Flight Recorder and that you do not enable the Query Log for the Analysis Services instance. You can disable these in the Analysis Server Properties dialog box for your Analysis Services instance. This server-side trace replicates most of the functionality of these two features. This Codeplex project includes scripts (in the Populating\_OLAP\_Query\_Log folder) that enable you to populate the OLAPQueryLog table with only the query data for queries that you identify as queries that will benefit from aggregations (rather than populating this table with information about every query). You can then use the Usage Based Optimizer to define aggregations based on the data in the OLAPQueryLog table. For more information about designing aggregations in SQL Server 2008 Analysis Services, see [Reintroducing Usage-Based Optimization in SQL Server 2008 Analysis Services](http://sqlcat.com/technicalnotes/archive/2008/11/18/reintroducing-usage-based-optimization-in-sql-server-2008-analysis-services.aspx).

## Analysis Services Performance Monitor Counters

This Data Collection solution uses Management Data Warehouse to collect Analysis Services Performance counters. By default, the System Data Collection Sets in Management Data Warehouse only collect general system information and information specific to the SQL Server 2008 relational engine. For more information about the SQL Server 2008 Management Data Warehouse, see [The Management Data Warehouse](http://msdn.microsoft.com/en-us/library/bb677306.aspx).

The **SSAS\_Monitoring\_Core** custom data collection set used in this solution contains a single collection item that utilizes a Performance Monitor collector type. This collection set records the core performance monitor data related to Analysis Services processing and query activity, as well as specific general system counters (I/O, memory, and processor). By using the Server Activity system collection set as a template (by generating a CREATE script from SQL Server Management Studio), defining this custom collection set was a relatively straight-forward task. We simply changed the collection set name, deleted the T-SQL collection item, renamed the Performance Counter collection item to SSAS\_Monitoring\_Core, and changed the XML list of Performance Monitor counters collected to the list of the counters that we wanted to collect for Analysis Services.

**Note**: For a full list of the counters collected or for information about how to modify the counters collected, see **Appendix A: Working With the Files**. We also defined a second collection set with all of the other Analysis Services performance monitor counters (for future reference), but we have not used this verbose collector in any of our tests.

This SSAS\_Monitoring\_Core collection set, by default, collects performance monitor data every 60 seconds, caches it locally using Integration Services raw file format and then uploads it every 15 minutes to the MDW database in the specified SQL Server instance. By default, collected data is retained indefinitely.

**Important**: There have been several bug fixes in the most recent Cumulative Update for SQL Server 2008 related to the robustness of the data collectors. As a result, we recommend that you install the most recent cumulative update when you use Management Data Warehouse collectors. Also, to minimize the performance impact, we recommend that you disable the System Data Collection Sets – unless you also wish to monitor the SQL Server instance on the same computer.

**Note**: You can configure Management Data Warehouse to purge data that it knows about after a specified number of days. This data is purged by using the sp\_purge\_data stored procedure. Management Data Warehouse does not know about the Profiler trace data or the Analysis Services DMVs. You will need to write your own mechanism to periodically selected purge this data, or modify this stored procedure. This Data Collection solution does include scripts to purge all data (using the Run\_Reset\_Collected\_Data.cmd batch file).

## Analysis Services Dynamic Management Views (DMVs)

SQL Server 2008 Analysis Services added Dynamic Management Views (DMVs) that enable you to view the state of the Analysis Services service at any point in time. This Data Collection solution, by default, uses an Integration Services package to periodically collect data from 6 DMVs for performance analysis. Alternatively, you can use a custom T-SQL collector with Management Data Warehouse to periodically collect DMV data. While we chose not to use this option, all of the code to utilize this option is included with this Codeplex solution. Both options are discussed in this section – each option collects and stores information in a slightly different manner and has advantages and disadvantages.

The DMVs queried in this solution can be split into two categories:

* DMVs returning performance data based on connection and session
  + **$system.discover\_connections** – this DMV provides resource usage and activity information about the currently opened connections on the server. For more information, see <http://msdn.microsoft.com/en-us/library/bb934105.aspx>.
  + **$system.discover\_sessions** – this DMV provides resource usage and activity information about the currently opened sessions on the server. For more information, see <http://msdn.microsoft.com/en-us/library/bb934101.aspx>.
  + **$system.discover\_commands** – this DMV provides resource usage and activity information about the currently executing or last executed command in each open connections on the server. For more information, see <http://msdn.microsoft.com/en-us/library/bb934103.aspx>.
  + **$system.discover\_command\_objects** – this DMV provides resource usage and activity information about the objects in use by the referenced command. For more information, see <http://msdn.microsoft.com/en-us/library/bb934100.aspx>.
* DMVs returning performance data based on time
  + **$system.discover\_object\_activity** – this DMV provides resource usage per object since the start of the service. For more information, see <http://msdn.microsoft.com/en-us/library/bb934094.aspx>.
  + **$system.discover\_object\_memory\_usage** – this DMV provides information about memory resources used by objects. For more information, see <http://msdn.microsoft.com/en-us/library/bb934098.aspx>.

### Collecting DMV Data Using Integration Services

The **Read\_SSAS\_DMV** Integration Services package contains 6 sequence containers, each of which uses an ADO.NET connection to connect to a specified Analysis Services instance and query a specific DMV and then record the result set in the MDW database in a specified SQL Server instance. The Integration Services package is called by the **SSAS\_Traces** SQL Agent job to periodically (by default every 60 seconds) to collect and upload the Analysis Services DMV data into the MDW database.

**Important**: We used the ADO.NET provider rather than native OLE DB provider for Analysis Services because we experienced periodic errors with OLE DB that we believe are related to the use of the unmanaged provider from within a .NET application. These errors only appeared when running the package using DTExec; they did not appear when the package was executed from within BIDS.

When querying for information about connections, sessions, and commands, we recorded the performance data related to each connection, session, and command in staging tables and then recorded the final state of information about each connection, session, or command in the permanent tables. This method enables us to quickly display the total I/O, memory and processor utilization for each connection, session, or command at query time rather than calculating these values at query time. This method does not enable us to see the progression of the use of these resources over the duration of a connection, session, or command (we took a different approach when querying DMVs using Management Data Warehouse that does enable us to see this progression). When querying for information about command objects, object activity and memory usage, we recorded the state of the information about each object at the time we queried it. This latter method enables us to view information about commands, object activity and memory as it changes over time – for example, to display it in a line chart.

### Collecting DMV Data Using Management Data Warehouse

As an alternative approach, we defined a custom collection set using the T-SQL Query collector type in Management Data Warehouse. With this approach, we defined a linked server between the SQL Server instance on which the collection set is running and the Analysis Services instance to be monitored. To query the Analysis Services DMVs discussed previously, we defined views for each DMV query containing OPENQUERY queries and then queried each view as separate queries in the custom T-SQL collector set in Management Data Warehouse. Using this approach, the state of each connection, session and command is written to the database each time the DMV is queries, which enables us to view the progression of the use of these resources over the duration of a connection, session, or command, but requires additional steps to report the maximum value for each of these resources for each connection, session or command.

**Note**: In the views, you will notice that we cast a number of the columns from the default data type of NTEXT to NVARCHAR due to limitations in the ability of the Management Data Warehouse collectors to support datatypes with more than 4000 characters.

There are numerous advantages to using the Management Data Warehouse framework to manage the data collection. These include a caching mechanism for superior performance, an integrated error handling and collector monitoring framework, and a set of system stored procedures for monitoring your collection sets. In a future version of this Codeplex project, we would like to convert the Integration Services packages in this solution to custom collectors. A white paper outlining the steps required to create a custom collector is expected soon from the product team.

# Installing the Data Collection Solution

Before installing, determine your data collection architecture. To minimize the performance impact on the Analysis Services instance being monitored, it is recommended that you run most of the data collection agents on and to collect all data to a centralized monitoring server (called the **MDWDataCollectionServer**) rather than on the Analysis Services instance (called the **SSAS Server**). To accomplish this, you must modify the data collection scripts as specified below. We have used command-prompt SET commands, SQLCMD and ASCMD environment variables, and Integration Services configuration files to simplify this customization.

Important: This Data Collection solution requires version 10.0.87.5 or newer of the ASCMD command line utility on the MDWDataCollectionServer. A compiled executable of version 10.0.87.5 for the x86 is included in this Codeplex project (this version will also work in the x64 environment). You can download the source code for the ASCMD command line utility from Codeplex at: <http://www.codeplex.com/MSFTASProdSamples/Release/ProjectReleases.aspx?ReleaseId=18652>.

## Using the Default Data Collection Mechanisms

To install the data collection solution, follow these steps:

1. Extract the files from the compressed file that you downloaded from Codeplex – the recommended location for the MonitoringSSAS folder in this compressed file in the root of your C: drive on the MDWDataCollectionServer.
2. Prepare the MDWDataCollectionServer:
   1. Install the SQL Server relational engine, Integration Services, and Reporting Services on the MDWDataCollectionServer. Configure SQL Agent to start automatically.
   2. Install the latest Cumulative Update for SQL Server 2008.
   3. If you are configuring a separate MDWDataCollectionServer and SSASServer, configure SQL Server for remote access by enabling the TCP/IP and/or Named Pipes network protocols.
   4. Create a folder on the MDWDataCollectionServer to store the Analysis Services trace data. The default location is C:\TraceFile on the MDWDataCollectionServer.
   5. Share this folder and ensure that the Analysis Services service account on the SSAS Server has write permissions to this share and NTFS write permissions to the folder.
   6. Verify that you can connect to the TraceFile share using [\\localhost\TraceFile](file:///\\localhost\TraceFile).

**Important**: The server-side trace connects to the TraceFile share using \\<MDWDataCollectionServer>\TraceFile. If you are running this data collection solution on a single machine with no network connections, you will need to configure the network loopback connector or change the Start\_SSAS\_Server\_Trace.xmla file to connect to the local file system (e.g. C:\TraceFile) rather than through the network card.

**Note**: If you are collecting from multiple SSAS Servers, you can either use a separate folder for each SSAS Server or a common folder. We have not tested with trace files from multiple Analysis Services instances – but this Data Collection solution was designed with this in mind.

1. Prepare the SSAS Server:
   1. Install the SQL Server relational engine and Analysis Services.
      1. Configure SQL Agent to start automatically.
      2. If you are collecting data to a remote MDWDataCollectionServer, ensure that the Analysis Services service account is using a domain user account with permission to write to the TraceFile share.
   2. On the SSAS Server, we recommend that you disable Flight Recorder and that you do not enable the OLAPQueryLog to minimize the performance impact of having two (or three) traces running.
2. Modify the SetVariables.cmd script in the MonitoringSSAS folder for your environment. This file is used during installation and execution to customize the solution components for your environment. Use the information in the following table to customize this script file.

**Important**: If you specify your server name as Localhost, the Reset scripts will fail as they attempt to delete data stored in the MDW database based on the server name.

**Note**: In most environments, you should not need to modify any other files.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Default Value** | **New Value** |
| MDWDataCollectionServerComputer | MDWDataCollectionServerComputerName | The name for your MDWDataCollectionServer |
| MDWDataCollectionServerInstance | MDWDataCollectionServerInstanceName | The name for the instance in which the MDW database is created on your MDWDataCollectionServer computer |
| SSASServerComputer | SSASServerComputerName | The name for your SSASServer |
| SSASServerDBInstance | SSASServerDBInstanceName | The name for the relational database instance on your SSASServer computer |
| SSASServerSSASInstance | SSASServerSSASInstanceName | The name for the SSAS instance on your SSASServer computer |
| DTEXEC\_PATH | C:\Program Files\Microsoft SQL Server\100\DTS\Binn\ | The path to the DTEXEC executable (either 32 or 64 bit) |
| DTEXECx86\_PATH | C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\ | The path to the DTEXEC executable (32 bit version – required to read the SQL Profiler trace data) |
| DataFilePath | C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\ | The path for the data files for the MDW and MDW\_Control databases. |
| MDWDataFile | MDW.mdf | The name for the main data file for the MDW database.  **IMPORTANT**: If the specified data file is in use by another SQL Server instance, this install script will fail. |
| MDW\_ControlDataFile | mdw\_control.mdf | The name for the main data file for the MDW\_Control database.  **IMPORTANT**: If the specified data file is in use by another SQL Server instance, this install script will fail. |
| LogFilePath | C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\ | The path for the log files for the MDW and MDW\_Control databases. |
| MDWLogFile | mdw\_log.ldf | The name for the log file for the MDW database.  **IMPORTANT**: If the specified log is in use by another SQL Server instance, this install script will fail. |
| MDW\_ControlLogFile | mdw\_control\_log.ldf | The name for the log file for the MDW\_Control database.  **IMPORTANT**: If the specified log is in use by another SQL Server instance, this install script will fail. |
| JobOwner | sa | Valid user name for the SSAS\_Trace and SSAS\_DMVs jobs on MDWDataCollectionServer |
| Load\_SSAS\_Trace\_Data\_Path | C:\MonitoringSSAS\Configuring\_Data\_Collection\_Server\ | The path for the SSAS\_Trace job on MDWDataCollectionServer |
| Trace\_File\_Path | C:\TraceFile | The path where you store the trace files. All files with an extension of trc in the specified path will be loaded by the SSAS\_Trace job. |
| SSASTraceFileSize | 100 | Maximum size in MB for each trace file |
| LoadTraceJobSchedule | 15 | The frequency in minutes for the execution of the SSAS\_Trace job |
| Read\_SSAS\_DMV\_Path | C:\MonitoringSSAS\Configuring\_Data\_Collection\_Server\ | The path for the SSAS\_DMVs job on MDWDataCollectionServer |
| ReadSSASDMVJobSchedule | 60 | The frequency in seconds for the execution of the SSAS\_Trace job |
| ReadSSASCoreJobSchedule | 60 | The frequency in seconds for the execution of the SSAS\_Trace job |
| PerfCounterPrefix | MSAS 2008 | The prefix for the SSAS performance counters. **IMPORTANT**: For named instances, the instance name must be entered entirely in capital letters. |
| ASMCD\_Path | C:\ASCMD\ | Path to version 10.0.87.5 or greater of the ASCMD command-prompt utility |
| MonitoringSSAS\_Drive | C: | The appropriate drive for your environment |
| MonitoringSSAS\_Path | \MonitoringSSAS\ | The path to the MonitoringSSAS folder |

**IMPORTANT**: Do not continue until you have modified all necessary script files.

1. Execute the **Configuring\_MDWDataCollectionServer.cmd** batch file in the MonitoringSSAS folder. This script calls a number of Transact-SQL scripts and Integration Services packages that do the following:
   1. Create the MDW database
   2. Create four schemas: MDW, SSAS, Stage, and Custom\_Snapshots
   3. Create objects in the MDW database for processing and storing Analysis Services trace data
   4. Load data into the EventClass and EventSubclass tables
   5. Create disabled job that loads Analysis Services trace data from the file system into the MDW database
   6. Create objects in the MDW database for storing Analysis Services DMV data
   7. Create a disabled job that collects DMV data into the MDW database
   8. Create the MDW\_Control database for storing metadata
   9. Create a control table in the MDW\_Control database
   10. Create a objects in the MDW\_Control database for querying the collected data
2. After the script completes, verify that all portions of this batch file complete successfully.
3. On the MDWDataCollectionServer, run the **Configure the Management Data Warehouse Wizard** to upgrade the MDW database that you created in step 5 for Management Data Warehouse. For detailed steps, see [How To: Configure the Management Data Warehouse](http://msdn.microsoft.com/en-us/library/bb933864.aspx).
4. On the SSAS Server, run the **Configure the Management Data Warehouse Wizard** to set up data collection of the Performance Monitor counters to the MDW database on the MDWDataCollectionServer.
5. Execute the **Configuring\_SSASServer.cmd** batch file in the MonitoringSSAS folder. This batch file calls a number of scripts that do the following:
   1. Stop the Management Data Warehouse system collection sets
   2. Purges any data that they have collected
   3. Creates the SSAS Monitoring-Core custom collection set
6. After the script completes, verify that all portions of this batch file complete successfully.

You are now ready to begin collecting Analysis Services performance data.

## Using a Custom Data Collector for SSAS DMVs

f you choose to use a T-SQL collector type in a custom collection set in Management Data Warehouse to collect DMV information, you will need to do the following:

1. If you choose to run this collector remotely, you will need to run step 8 on the MDWDataCollectionServer.
2. You will also need to modify the install scripts to remove the reference to the **Create\_Objects\_For\_SSAS\_DMV\_MDW.sql** script in the **Configuring\_MDWDataCollectionServer.cmd** batch file.
3. On the server on which this collector will be running, you will need to run the **SSAS\_DMVs.sql** script in the C:\MonitoringSSAS\Configuring\_Data\_Collection\_Server\Collecting\_SSAS\_DMV\_Data\_TSQL\_Collector folder.
4. On the server on which this collector will be running, you will need to modify and run the **AddLinkedServer.sql** script in the C:\MonitoringSSAS\Configuring\_Data\_Collection\_Server\Collecting\_SSAS\_DMV\_Data\_TSQL\_Collector folder.
5. On the server on which this collector will be running, you will need to modify and run the **DMV\_Views\_OpenQuery.sql** script in the C:\MonitoringSSAS\Configuring\_Data\_Collection\_Server\Collecting\_SSAS\_DMV\_Data\_TSQL\_Collector folder.

# Starting

When you are ready to start collecting Analysis Services performance data, the following steps are required:

1. Ensure that SQL Server agent is running on SSAS Server and MDWDataCollectionServer.
2. Execute the **Start\_Collection.cmd** batch file in the SSASMonitoring folder.
3. Verify that the SSAS\_DMVs job, the SSAS\_Trace job and the SSAS\_Monitoring\_Core collection set are running.
4. Periodically review the job history and collection set logs.

# Stopping Data Collection

When you are ready to stop collecting data, execute the **Stop\_Collection.cmd** batch file in the SSASMonitoring folder. This batch file stops the server-side trace, runs the SSAS\_Trace job, disables the SSAS\_Traces and SSAS\_DMVs jobs, stops the SSAS\_Monitoring\_Core collection set (uploading any cached data to MDWDataCollectionServer) and runs the load\_perfmon\_counter\_config\_table stored procedure.

# Resetting Data Collection

If you wish to reset data collection by purging all collected data in the MDW and MDW\_Control databases, execute the **Reset\_Collected\_Data.cmd** in the SSASMonitoring folder. This batch file calls three Transact-SQL scripts that truncate the data in the MDW database.

# Periodically Truncating Old Data

By default, the data collected in the Management Data Warehouse by this solution is not purged automatically. You will need to establish your own purge or truncation schedule. The amount of data collected over time can be quite substantial.

# Using the Sample Reports to Query the Collected Data

This Codeplex project includes 12 sample reports that query the collected data. These reports rely on two tables (MDW.Workload\_Runs and MDW.Perfmon\_Counter\_Config) and 12 stored procedures in the MDW\_Control database. These tables and stored procedures and tables are created by the as part of the installation of the MDWDataCollectionServer. The use of these objects is discussed below.

We developed the sample reports to help us analyze multiple workload runs. For example, we have used these reports to compare query and processing performance between multiple runs changes either hardware, such as more memory or faster I/O subsystem, or changing Analysis Services configuration properties. You could also use these reports to compare a specified time period with a baseline. We have also used these reports to analyze an Analysis Services profiler traces sent to us by customers to determine the worst performing queries and to quickly identify the types of problems that the data reveals are the source of the problem for the poorly performing queries.

After you have collected data, this section discussed how to use the sample reports in this project to analyze your data.

## Populating the MDW.Perfmon\_Counter\_Config Table

The MDW.Perfmon\_Counter\_Config table contains the Performance Monitor counters for each server for which you have collected data and a value (Y/N) indicating whether you wish to include this counter in your reports. This table is populated by the load\_perfmon\_counter\_config\_table stored procedure. In the C:\MonitoringSSAS\Stopping\_Data\_Collection folder, there is a script for executing this stored procedure and populating the perfmon\_counter\_config.

**Important**: You must run this stored procedure before attempting to run any of the reports. If you do not, you will receive an error stating that the MDW.perfmon\_counter\_config object is invalid. This stored procedure is executed by the Stop\_Collection.cmd batch file.

## Populating the MDW.Workload\_Runs Table

We use the MDW.Workload\_Runs table to store the necessary information about each workload run. The C:\MonitoringSSAS\Analyzing\_SSAS\_Perf\_Data\_on\_Monitoring\_Server\Scripts folder contains a sample script for inserting data into this table.

## Installing the Sample Reports

The C:\MonitoringSSAS\Analyzing\_SSAS\_Perf\_Data\_on\_Monitoring\_Server folder contains the SSAS\_Reports solution. The reports all use a shared data source, called MDW\_Control, which connects to the MDW\_Control database with a data source of Localhost. The project is configured to deploy the reports to the SSAS\_Reports folder on your local server and the data source to the Data Sources folder (and to overwrite the data source at the destination). Modify these settings as appropriate and deploy the entire project to the Report Server of your choice.

## Running the Sample Reports

You can use these sample reports to analyzing MDX query workloads or Analysis Services processing performance.

### Analyzing Queries

We have included several sample reports that analyze queries. The figures used show the data from nine query stream that are executed against the Adventure Works 2008 Analysis Services sample database (called the Sample Workloads).

* Run 1: A time period that encompasses the time periods for Runs 2-6 – an elapsed time of approximately 14 minutes.
* Run 2: A time period that encompasses the time period required to execute all 9 query streams in serial, while no processing operations are occurring – an elapsed time of approximately 3 minutes.
* Run 3: A time period that encompasses the time period required to execute all 9 query streams in parallel, while no processing operations are occurring – an elapsed time of approximately 1 minute.
* Run 4: A time period that encompasses the time period required to execute all 9 query streams in serial, while the Adventure Works cube is fully processed – an elapsed time of approximately 3 minutes.
* Run 5: A time period that encompasses the time period required to execute all 9 query streams in parallel, while the Adventure Works cube is fully processed – an elapsed time of approximately 1.5 minutes.
* Run 6: A time period that encompasses the time period required to fully process the Adventure Works cube – an elapsed time of approximately 1 minute.

For these workloads, we collected data every 15 seconds. For the first workload, we used ASCMD to simultaneously execute a text file containing 3 related, but different queries (representing a query stream) on separate connections. We executed this query stream three times on each of the separate connections, clearing the cache between each workload execution. For the second workload, we executed the identical queries in the same manner, but also performed a full process on the Adventure Works cube during each execution of the query stream (for a total of three process operations). For the third workload, we executed the full process of the Adventure Works cube three times while no queries were being executed.

**Note**: A backup of the results of the workloads is included in this Codeplex project (backups of the MDW and MDW\_Control databases, ASCMD scripts, and a Microsoft Excel spreadsheet containing the output from the query workload runs).

* **MDX Querying – Workload Comparison Summary** – This sample report analyzes Analysis Services trace data for one or more query workloads, with a workload being defined as all queries that execute during a particular time period. The report allows you to limit the total number of queries returned, specify the duration and CPU time thresholds, and specify the number of times a query must execute before it is included in the report. The report includes the following items for the queries in each workload:
  + The workload name
  + The workload description
  + The number of queries in each workload
  + The total duration for all of the queries in each workload
  + The total storage engine duration for all of the queries in each workload
  + The total formula engine duration for all of the queries in each workload
  + The total CPU time for all of the queries in each workload
  + The total number of measure groups for all of the queries in each workload
  + The total number of non-cached subcubes for all of the queries in each workload
  + The total number of cached subcubes for all of the queries in each workload
  + The total number of partitions queried for all of the queries in each workload
  + The total number of aggregations used for all of the queries in each workload

This sample report also includes the following links that enable you to perform more analysis (the details of each report is described later in this document):

* + **Link to Perf Monitor Data** – This is a link to the CountersDataPoints report, which enables you to view selected performance monitor counters for the time periods represented by the workloads. This report is discussed later in this section.
  + **Link to AS DMV Data By Time** – This is a link to the DMV Session Info By Time report, which enables you to view all DMV activity for the time periods represented by the workloads. This report is discussed later in this section.
  + **Link to Workload Performance Detail** – This is a link to the MDX Querying – Workload Comparison Detail report, which enables you to drill into more trace file detail for each of the queries executed during the time periods represented by the workloads. This report is discussed later in this section.

Figure 1 illustrates the execution of the MDX Querying – Workload Comparison Summary report for the Sample Workloads showing the summary of query execution performance for the time period defined by Run 1, regardless of count, duration, or CPU time. Notice that the formula engine was the primary bottleneck for these queries.

**Note**: Analysis Services trace data does not enable you to determine the cube name for queries that are resolved entirely from data already stored in the formula engine – we designate the cube name for these queries as FE\_Cache.

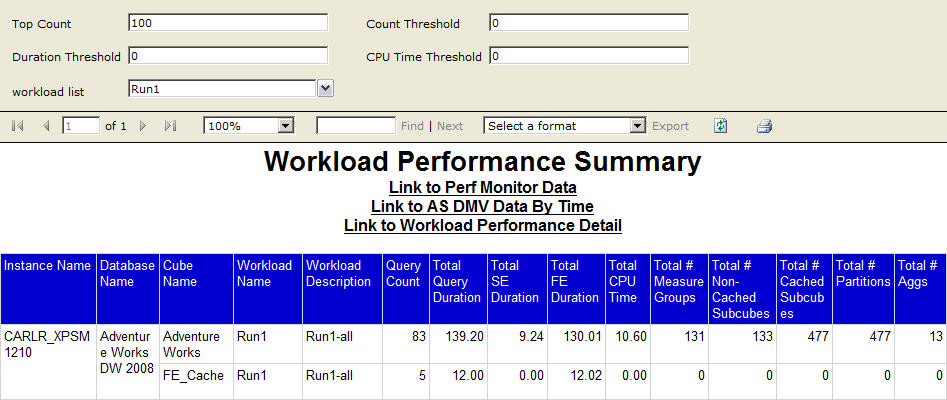


Figure 1: MDX Querying – Workload Comparison Summary report showing all queries for Run1

Figure 2 illustrates the execution of the MDX Querying – Workload Comparison Summary report for the Sample Workloads the summary of query execution performance for the time periods defined by Runs 2, 3, 4 and 5 for queries whose total duration was more than 1 seconds for at least one execution of the query, regardless of count or CPU time. Notice the following:

* + Runs 3 and 5 took substantially longer to complete than Runs 2 and 4. Executing all 9 query streams in parallel required more time per query, even though the elapsed time to run all of the queries was significantly less when the query streams were executed in parallel.
  + Performing the processing operation during Run 5 impacted the performance of parallel execution, while performing the same processing operation during Run 4 had only a modest impact on performance of serial execution. Notice in particular that there was one query during Run 5 that was resolved entirely from formula engine cache, but that it still took 11.40 seconds to complete – due either to contention from the processing operation or an inefficiently written query (one that the formula engine was unable to optimize).
  + While the formula engine was a bottleneck for all runs, the impact of insufficient processor resources was more significant during parallel execution and most dramatic during parallel query execution while processing was occurring.
  + The storage engine performance was also impacted by parallel query execution and by the processing operation.
  + The reuse of cache from previously executed queries and the clearing of the cache by the processing operation during the query runs is evidenced by the differing number of measure groups, non-cached and cached subcubes, partitions, and aggregations touched for the different runs.

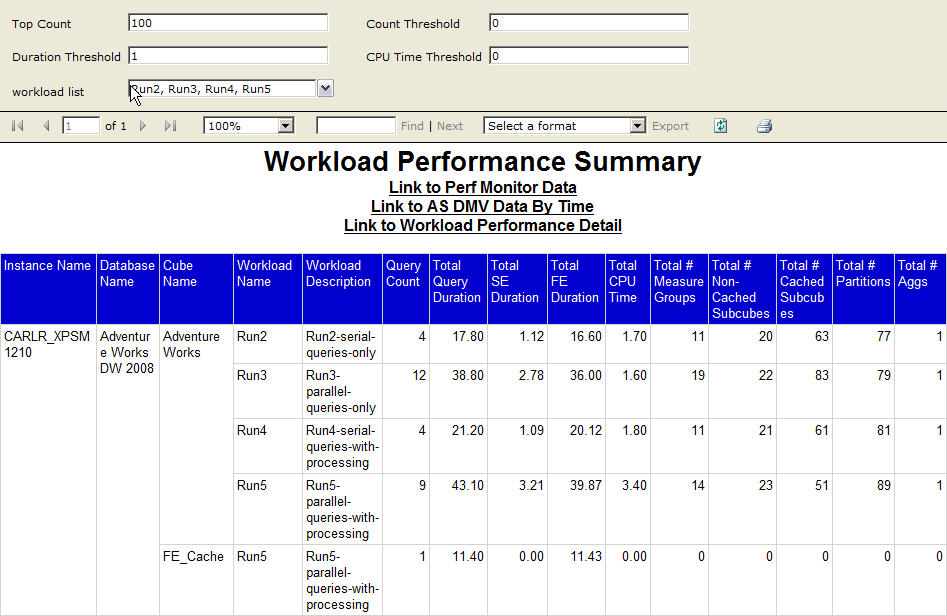


Figure 2: MDX Querying – Workload Comparison Summary report showing long-running queries for Runs 2- 5

By drilling through to the MDX Querying - Workload Comparison Detail report, we can look at the slow queries in more detail. For example, we will want to know if some queries are always slow and if some are only slow due to contention for system resources. The performance tuning techniques are obviously different depending upon the reason for the slow performance.

* **MDX Querying - Workload Comparison Detail** – This sample report analyzes Analysis Services trace data for the queries in one or more query workloads, comparing the performance of each identical query in each workload. The report allows you to limit the total number of queries returned, specify the duration and CPU time thresholds, and specify the number of times a query must execute before it is included in the report. The report includes the following items for each query in each workload:
  + The query ID
  + The workload names in which each unique query appeared
  + The workload descriptions for each workload in which each unique query appeared
  + The execution count for each unique query in each workload
  + The average total duration for each unique query in each workload
  + The average storage engine duration for the unique queries in each workload
  + The average storage engine duration percentage of the total average query duration for the unique queries in each workload
  + The average formula engine duration of the unique queries in each workload
  + The average formula engine duration percentage of the total average query duration for the unique queries in each workload
  + The average CPU time for the unique queries in each workload
  + The average number of measure groups for the unique queries in each workload
  + The average number of non-cached subcubes for the unique queries in each workload
  + The average number of cached subcubes for the unique queries in each workload
  + The average number of partitions for the unique queries in each workload
  + The average number of aggregations for the unique queries in each workload

**Note**: You could modify this report to return all queries during a specific time period that exceeded some specified threshold – and use the new gauge controls to embed this information into a performance monitoring dashboard.

This sample report also includes links that enable you to perform more analysis (the details of each report is described later in this document):

* + **Link to Perf Monitor Data** – This is a link to the CountersDataPoints report, which enables you to view selected performance monitor counters for the time periods represented by the workloads. This report is discussed later in this section.
  + **Link to AS DMV Data by Time** – This is a link to the DMV Session Info By Time report, which enables you to view all DMV activity for the time periods represented by the workloads. This report is discussed later in this section.
  + **Link by Query ID** – This is a link from a specified query ID to the MDX Querying – Multiple Executions of this Query report, which enables you to drill into more trace file detail for all the executions of a specified query executed during the selected time periods represented by the workloads. This report is discussed later in this section.

Figure 3 illustrates the execution of the MDX Querying - Workload Comparison Detail report for the Sample Workloads showing queries whose total duration was more than 2.5 seconds for at least one execution of the query in each workload, regardless of count, duration, or CPU time. Notice the following:

* + All queries varied in performance between executions – which means that, if you were troubleshooting performance for these workloads, you should focus on what is competing for server resources during their execution.
  + Query ID 10 was the slowest performing query during all runs, was the slowest during Run 5, and was entirely formula engine bound for both executions. Notice also that Query ID 10 was the query that was slow when it was resolved entirely from cache. This information definitely tells us that this query might benefit from a more efficient query design. Indeed, Query ID 15 is the optimized version of this same query – which takes less than one-tenth of a second to execute. The optimization of this query is discussed in [Ranking in MDX](http://www.sqljunkies.com/WebLog/mosha/archive/2006/03/14/mdx_ranking.aspx).
  + Query ID 8 was both formula engine and storage engine bound during Runs 3 and 5, but executed in less than 2 seconds in Runs 1 and 3. This query was significantly impacted in formula engine performance during Run 5 due to the processing operation, and was modestly impacted in storage engine performance – either from competition for disk resources and/or (more likely given the data in this report) due to the clearing of storage engine caches at the end of the processing operation – as evidenced by increased number of non-cached subcubes and partitions touched (and the decreased number of cached-subcubes). To determine what is competing with Query ID 8 for resources during Runs 3 and 5, we will drill into the executions of QueryID 8 (MDX Querying - Multiple Executions of a Specific Query) and drill into performance counters for each execution (the CountersDataPoints report).

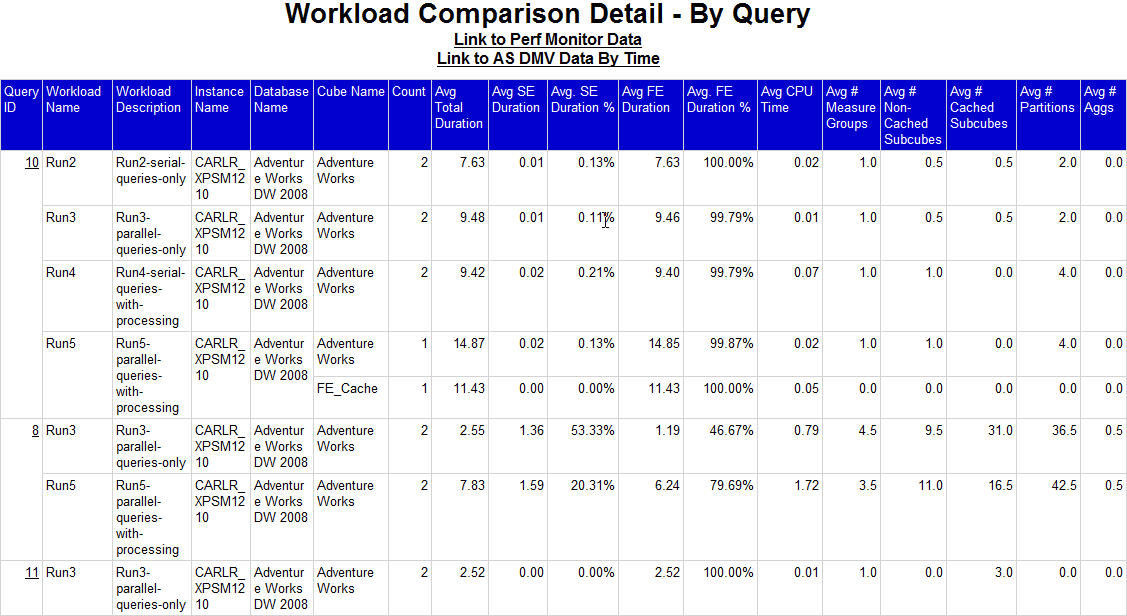


Figure 3: MDX Querying - Workload Comparison Detail report showing long-running queries during each run

* **MDX Querying - Multiple Executions of a Specific Query** –This sample report analyzes Analysis Services trace data for a specified query in one or more query workloads, comparing the performance of each execution of the query in each workload – ordered by Total Duration. The report includes the query text and the following items for each query in each workload:
  + The query ID
  + The workload names for each execution of the specified query
  + The workload descriptions for each workload for each execution of the specified query
  + The execution ID for each execution of the specified query
  + The begin time for each execution of the specified query
  + The end time for each execution of the specified query
  + The total duration for each execution of the specified query
  + The storage engine duration for each execution of the specified query
  + The storage engine duration percentage of the total query duration for each execution of the specified query
  + The formula engine duration for each execution of the specified query
  + The formula engine duration percentage of the total query duration for each execution of the specified query
  + The CPU time for each execution of the specified query
  + The number of non-cached subcubes for each execution of the specified query
  + The number of cached subcubes for each execution of the specified query
  + The number of partitions for each execution of the specified query
  + The number of aggregations for each execution of the specified query

The MDX Querying – Workload Comparison Detail links to this report. This sample report also includes links that enable you to perform more analysis (the details of each report is described later in this document):

* + **Link to Perf Monitor Data** – This is a link to the CountersDataPoints report, which enables you to view selected performance monitor counters for the time periods represented by the workloads. This report is discussed later in this section.
  + **Link to AS DMV Data by Time** – This is a link to the DMV Session Info By Time report, which enables you to view all DMV activity for the time periods represented by the workloads. This report is discussed later in this section.
  + **Link to Trace Detail for a Specific Query** – This is a link for to the MDX Querying – Multiple Executions of this Query report via the Storage Engine Duration column or the SE Duration % column, which enables you to drill into the trace file for storage engine detail regarding the specific execution of a query during the selected time periods represented by the workloads. This report is discussed later in this section.
  + **Link To DMV Detail for a Specific Query** – This is a link from a specified query execution ID to the DMV Session Info By Session report from the Conn. ID column, Session ID column, or SPID column, which enables you to drill into DMV detail for this particular query execution by ConnectionID, SessionID and SPID during the selected time periods represented by the workloads. This report is discussed later in this section.

Figure 4 illustrates the execution of the MDX Querying - Multiple Executions of a Specific Query report for the Sample Workloads for QueryID 8, the query that was storage engine and formula engine bound in Runs 3 and 5. As you saw in the previous report, there were 2 executions of this query that took an average of 2.55 seconds in Run 3 and there were 2 executions of this query that took an average of 7.83 seconds in Run 5. In this report, notice the following:

* + Query Execution ID 88 in Run 5 was the slowest execution – although this execution of the query touched less measure groups and partitions and used less cached and non-cached subcubes. In particular, notice that this execution required the least storage engine resources and the most formula engine resources. In the absence of contention for resources, this query should have executed faster than the other two executions displayed in this report.
  + Query Execution IDs 83 and 37 have a very similar execution signature in the trace data – both are essentially resolved from cold cache and touch the same number of measure groups, partitions, and aggregations. They also use the same number of cached and non-cached subcubes. When looking at these executions of these queries, notice that the storage engine time is approximately 2.7 seconds and that only 1 aggregation was used for each query. Based on this information, you can see that this server might benefit from a faster disk subsystem – or more aggregations if the cube design supported more aggregations (because the Adventure Works cube uses measure expressions for currency conversion, aggregations cannot be used on those measure groups with measure expressions).

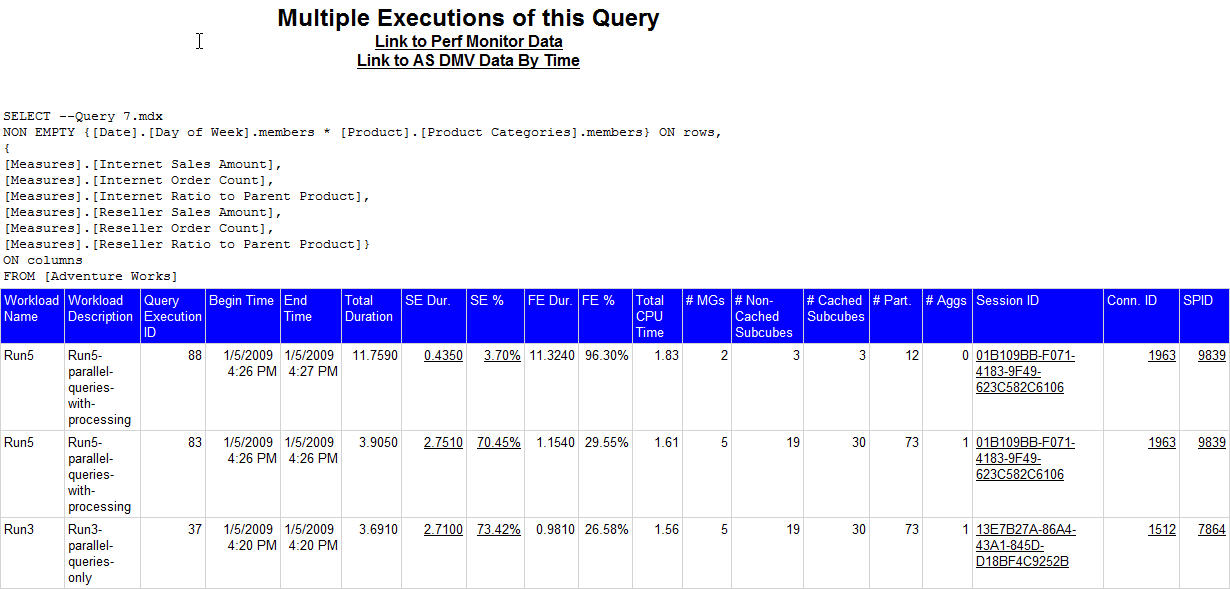
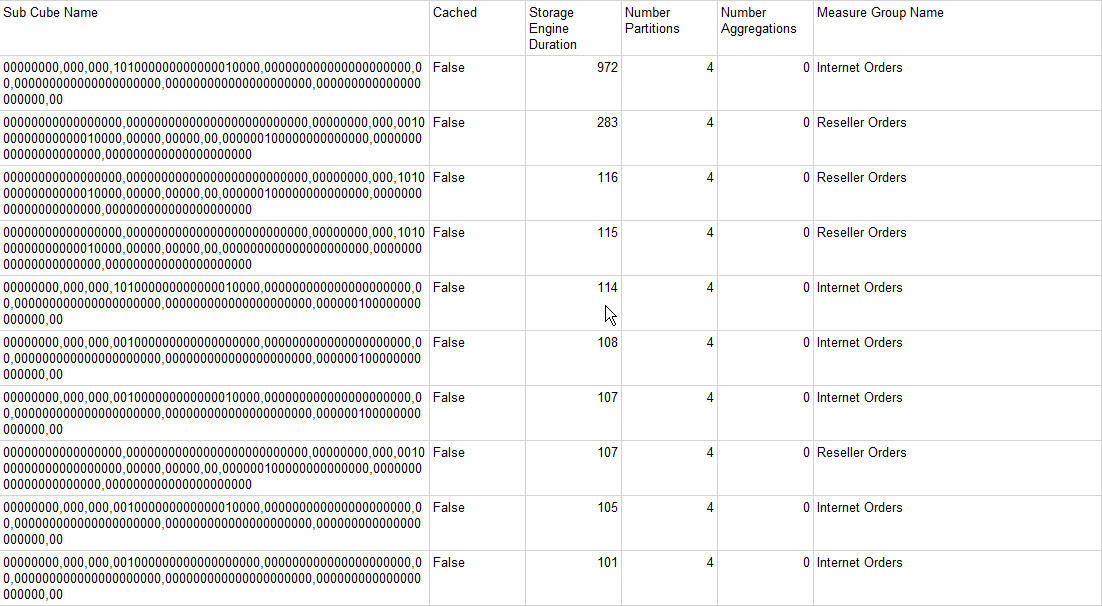


Figure 4: MDX Querying - Multiple Executions of a Specific Query (QueryID 8)

* **MDX Querying – Details of the Execution of a Specific Query** –This sample report analyzes Analysis Services trace data for detailed query information regarding the storage engine execution with respect to a specific execution of a specific query – ordered by Storage Engine duration. The report includes the following items for the specific execution of the query, sorted by duration:
  + The subcube name – this is the subcube vector, which you can use to build custom aggregations
  + Whether the subcube was cached or not
  + The storage engine duration for the subcube, as viewed from the formula engine
  + The number of partitions touched by the storage engine to resolve the subcube request
  + The number of aggregations touched by the storage engine to resolve the subcube request
  + The measure group to which the subcube relates

The MDX Querying - Multiple Executions of a Specific Query links to this report. This sample report includes no additional links.

Figure 5 illustrates the execution of this report for the Sample Workloads for Query Execution ID 83, the execution of QueryID 8 with the highest number of non-cached subcubes requests and the most partitions. Notice that the measure groups with the distinct count measure groups are the measure groups that required the most time of the storage engine and that an aggregation was used only for the Fact Currency Rate partition. For more information on aggregations, distinct counts and performance, see [SQL Server 2008 Analysis Services Performance Guide](http://sqlcat.com/msdnmirror/archive/2008/11/14/sql-server-2008-analysis-services-performance-guide-now-available.aspx) and [Analysis Services Distinct Count Optimization](http://sqlcat.com/whitepapers/archive/2008/04/17/analysis-services-distinct-count-optimization.aspx).



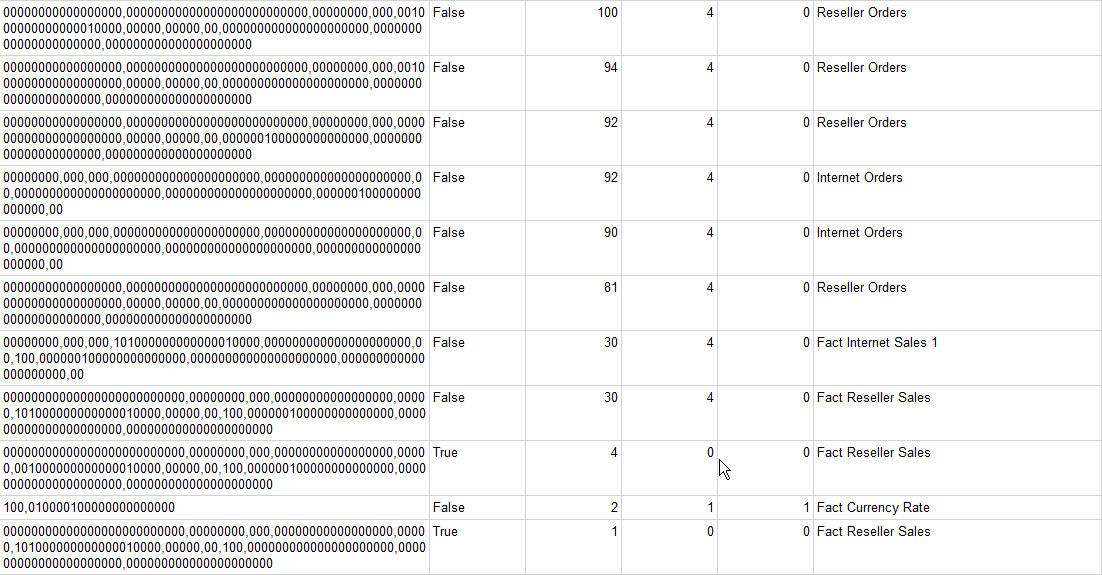


Figure 5: MDX Querying - Details of the Executions of a Specific Query (Query Execution ID 83)

* **CountersDataPoints** – This sample report analyzes Performance Monitor data collected by the SSAS\_Monitoring\_Core data collection set in Management Data Warehouse and compares the performance monitor values for selected counters for each selected workload. The get\_counter\_data\_points\_by\_config stored procedure that is used by this report uses a zero-based elapsed time calculation to overlay the data from each counter for multiple time periods on the line charts used by this report.

The MDX Querying – Workload Comparison Summary, MDX Querying - Workload Comparison Detail, MDX Querying - Multiple Executions of a Specific Query, DMV Session Info By Time and DMV Session Info By Session reports all link to this report. This report links to no other reports.

Figures 6, 7, 8, 9 and 10 illustrate the execution of this report for the Sample Workloads.

* + Figure 6 shows the counter names that are available for analysis based on the SSAS\_Monitoring\_Core data collection set.
  + Figure 7 shows the Avg time/query object counter for the MSAS 2008: Storage Engine Query object. Notice that the average time per query peaks at about 1.5 seconds for Runs 3 and 5, while it remains well below 1 second for Runs 2 and 4.
  + Figure 8 shows the C: counter for the Avg. Disk Queue Length Performance Object. Notice that disk queuing occurs during both Runs 3, 4 and 5.
  + Figure 9 shows the 0 and 1 counters for the % Processor Time Performance Object. Notice that Runs 3 and 5 requires more processor resources than either Runs 2, 4 or 6.
  + Figure 10 shows the Rows read/sec and Rows written/sec for the MSAS 2008: Processing object. Notice that the rows read/sec varies somewhat between runs, with Run 5 having the lowest rows read/sec. Notice also that the rows written/sec varies only slightly between runs. In general, for this simple test, processing performance is not substantially affected by the query runs – rather, the query runs are more affected by processing.

Note: Since the SSAS\_Monitoring\_Core data collection set only collects data every 15 seconds, data points at various points in querying or processing may not be sampled – such as at the beginning or end of a processing operation.

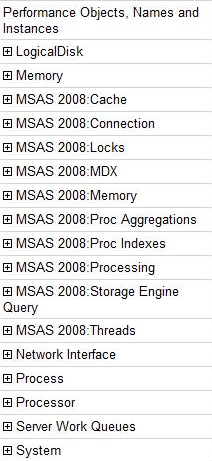


Figure 6: CountersDataPoints – Performance Objects Before Expansion of Selected Objects

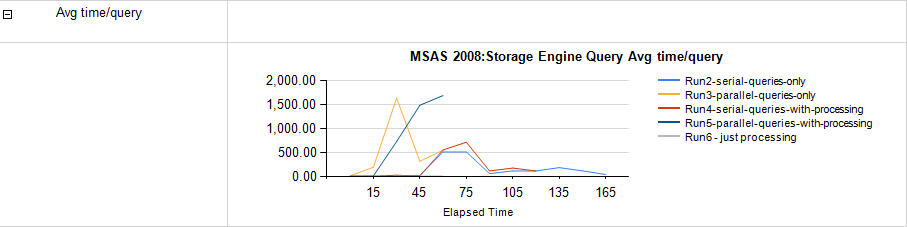


Figure 7: CountersDataPoints – MSAS 2008: Storage Engine Query Avg time/query

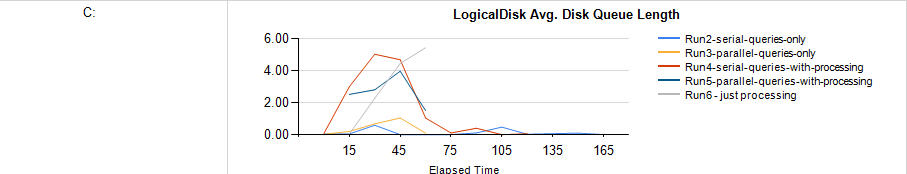


Figure 8: CountersDataPoints – Avg. Disk Queue Length – C:

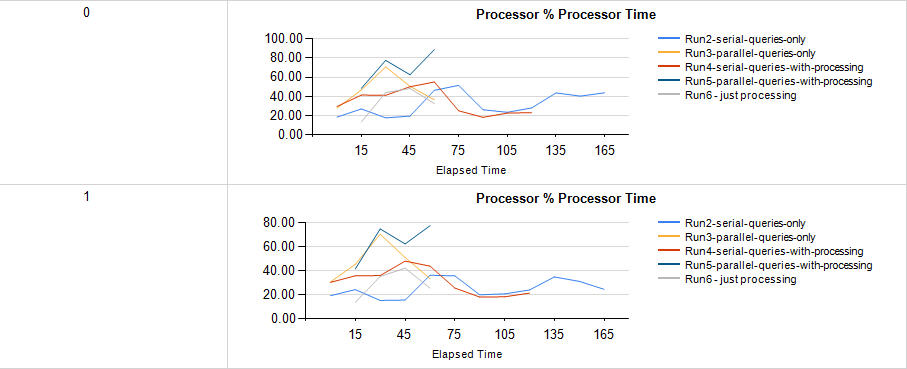


Figure 9: CountersDataPoints – % Processor Time –0 and 1

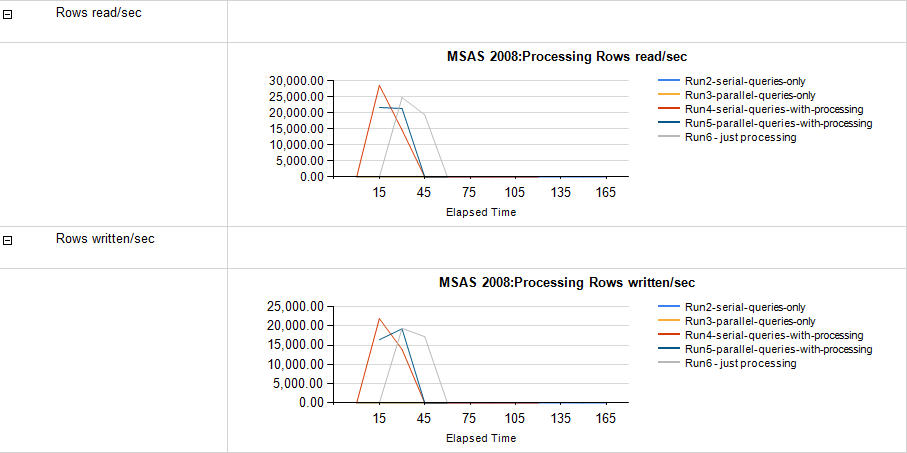


Figure 10: CountersDataPoints – MSAS 2008: Processing – Rows read/sec and Rows written/sec

* **DMV Session Info By Time** – This sample report analyzes Analysis Services DMV data and returns information about all Analysis Services activity during the time periods defined by each workload. You can use this report to determine other activity on a server during a specified time period that might affect query or processing performance. You can also use this report to compare the performance of different executions of a query workload or a processing operation.

The report includes the following items for each session – sorted by Session\_CPU\_Time\_MS:

* + Session ID – you can correlate this information with query or processing information collected by SQL Server Profiler in the Analysis Services trace data – these sample reports only correlate from trace data into the DMV data based on time, not session ID.
  + Workload name
  + Elapsed Time (ms)
  + CPU Time (ms)
  + Reads
  + Read (kb)
  + Writes
  + Write (kb)
  + Last Command – for query sessions, this column displays the last query. However, for processing sessions, this column displays no information.

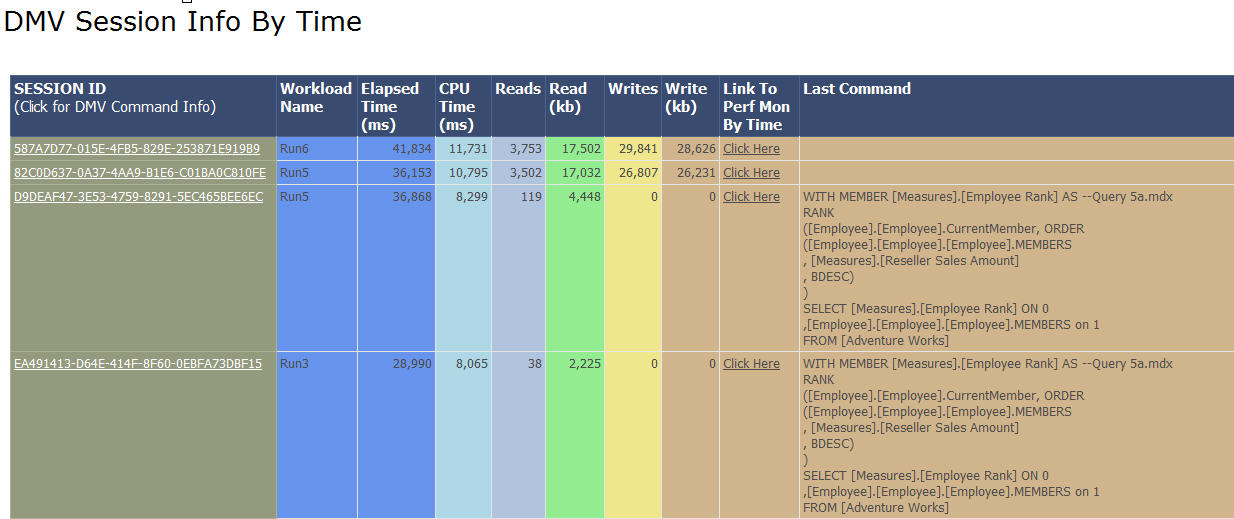
The MDX Querying – Workload Comparison Summary, MDX Querying - Workload Comparison Detail, MDX Querying - Multiple Executions of a Specific Query, and DMV Session Info By Session reports all link to this report. This sample report also includes links that enable you to perform more analysis (the details of each report is described later in this document):

* + **Link to Perf Mon By Time** – This is a link to the CountersDataPoints report, which enables you to view selected performance monitor counters for the time period represented by a specific workload. This report was discussed earlier in this section.
  + **Link to DMV Command Info** – This is a link to the DMV ObjectDetail\_PerSession report, which enables you to view the objects touched by a specific session. This report is discussed later in this section.

Figure 11 illustrates the execution of this report for Runs 5 and 6 for the Sample Workloads. Notice the following:

* + The top 2 most expensive sessions are processing sessions – measured by CPU Time, Reads, Read (kb), Writes and Write (kb). Notice that processing while no queries were running (Run 6) appears to be more expensive and last longer than processing while queries were running (Run 5). However, since the sample interval is 15 seconds – we can safely say that Runs 5 and 6 are similar in their performance characteristics but we cannot safely say that Run 6 was more expensive than Run 5.
  + Query 5a was the most expensive query, as discussed previously (Query ID 8) – and that it was slowest in Run 5 (with processing) compared to Run 3 (without processing).
  + Query 7 was the next most expensive query, as discussed previously (Query ID 10) – and that it was slowest in Run 5 (with processing) compared to Run 3 (without processing).

**Note**: When analyzing Runs 2 and 4, you need to be aware that when executing many queries in single session, you can analyze the session using DMVs but will find it difficult to analyze individual queries within a session. You can however, view the last command in a session.



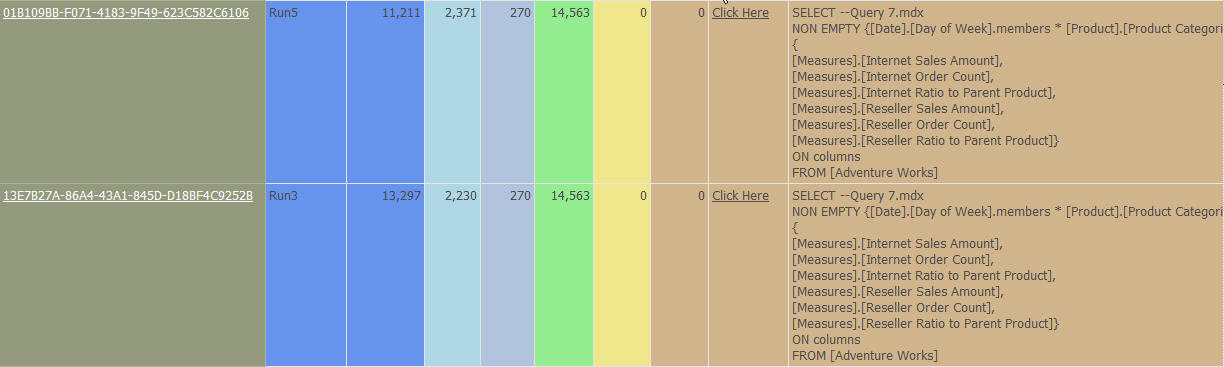


Figure 11: DMV Session Info By Time

* **DMV Session Info By Session** – This report analyzes Analysis Services DMV data and returns information about specific ConnectionID, SessionID and SPID information. Not every session that appears in the Analysis Services Profiler trace will appear in the DMV info – because DMV information is only being collected periodically. You can use this report to determine resource usage by a particular session. The report includes the following items for each session:
  + Start time
  + End time
  + Workload name
  + Elapsed Time (ms)
  + CPU Time (ms)
  + Reads
  + Read (kb)
  + Writes
  + Write (kb)
  + Last Command – for query sessions, this column displays the last query. However, for processing sessions, this column displays no information.

The MDX Querying - Multiple Executions of a Specific Query report links to this report. This sample report also includes links that enable you to perform more analysis (the details of each report is described later in this document):

* + **Link to Perf Mon By Time** – This is a link to the CountersDataPoints report, which enables you to view selected performance monitor counters for the time period represented by a specific workload. This report was discussed earlier in this section.
  + **Link to AS DMV Data by Time** – This is a link to the DMV Session Info By Time report, which enables you to view all DMV activity for the time periods represented by a specific workload. This report is discussed earlier in this section.

Figure 12 illustrates the execution of this report for the Sample Workloads for Query Execution IDs 88 and 83 of QueryID 10 (both were executed in the same session). From this report, you can link to the DMV Session Info By Time and CountersDataPoints reports to determine other activity on the server while this session was executing.

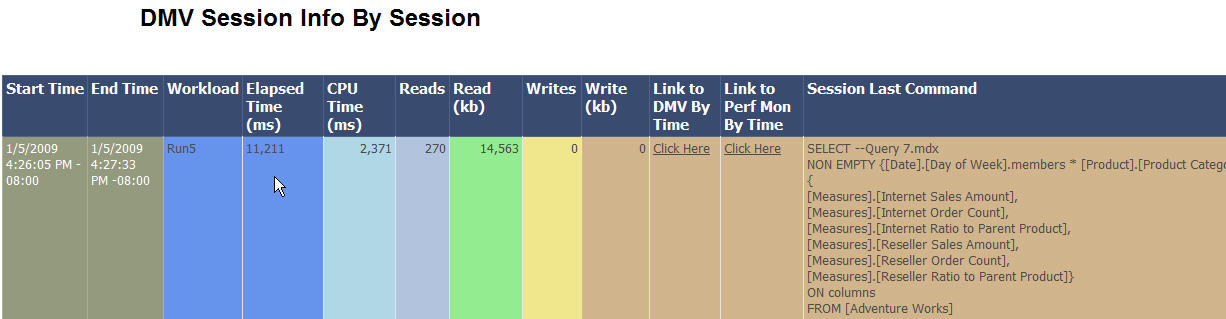
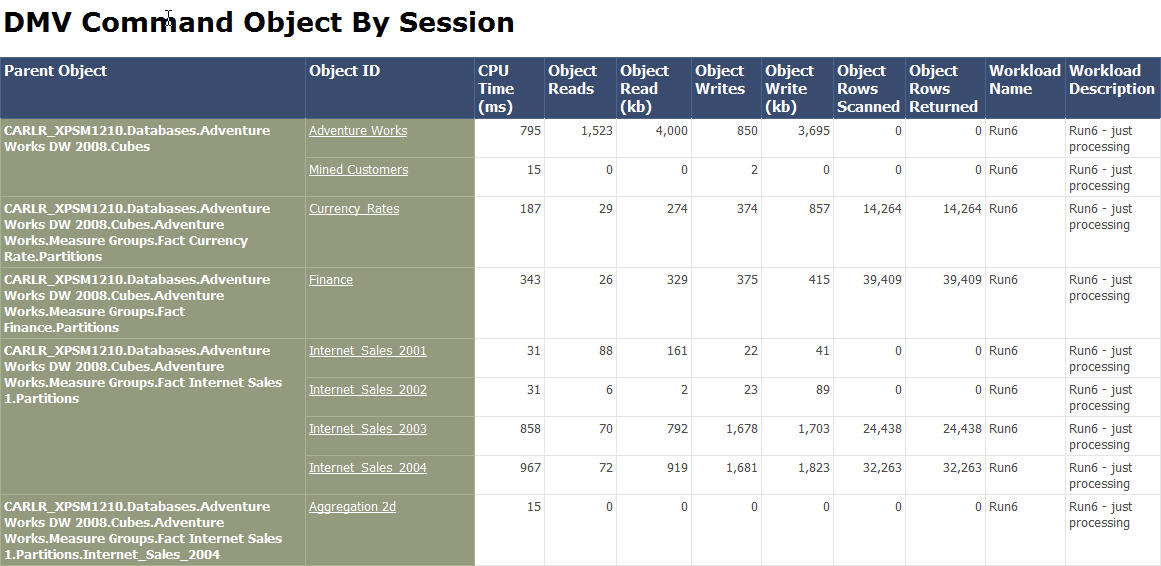


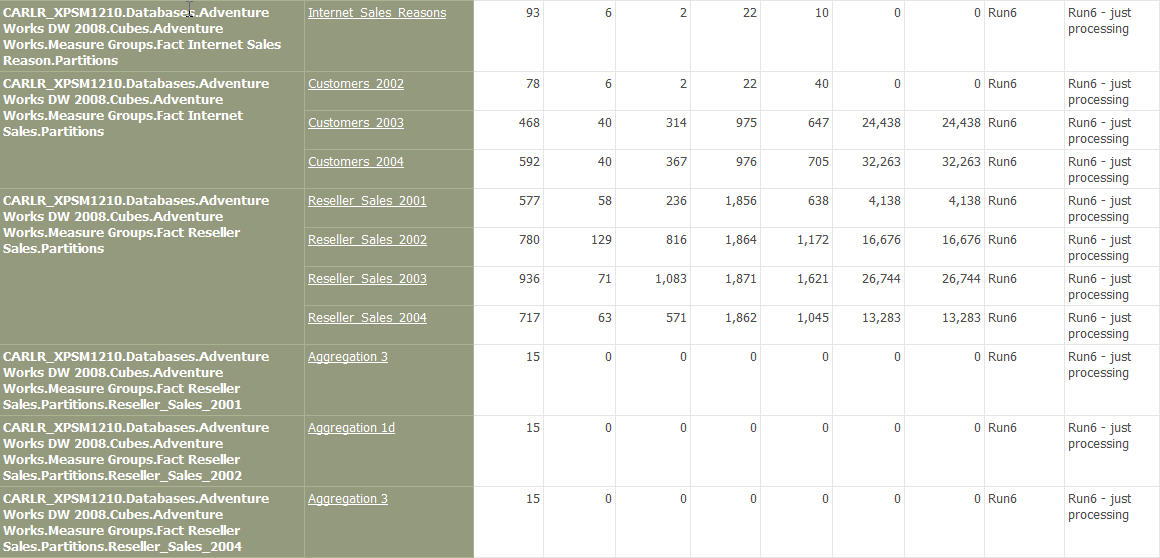
Figure 12: DMV Session Info By Session

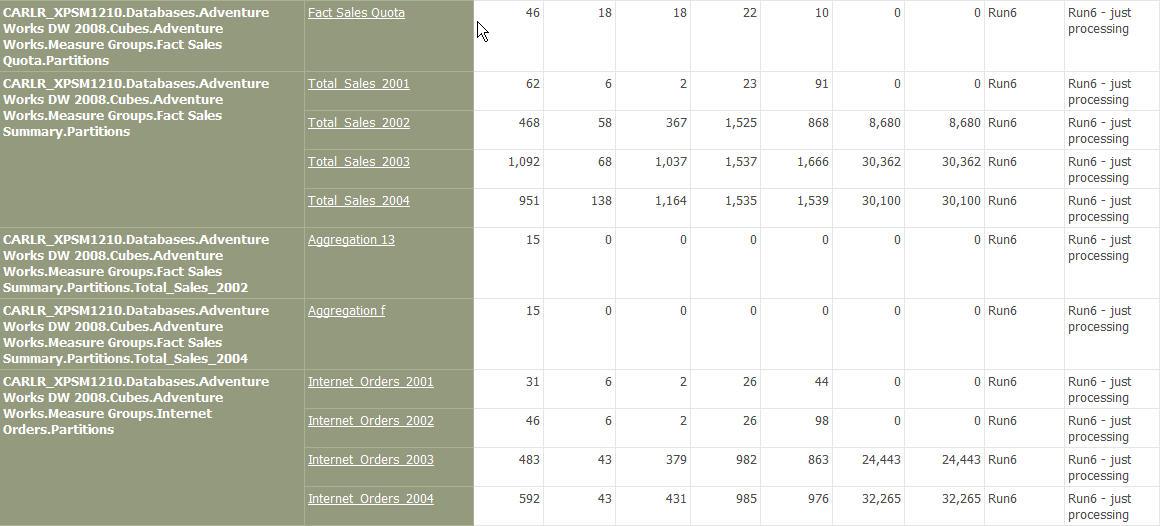
* **ObjectDetail\_PerSession** – This sample report analyzes Analysis Services DMV data and returns information about object activity for a particular session. You can use this report to determine resource usage by object for a particular session. For example, you can determine the resource utilization by partition for a particular query session or the objects that required the most resources during a processing operation. The report includes the following items for each session – grouped by Parent Object and ObjectID:
  + Parent Object
  + ObjectID
  + CPU Time (ms)
  + Object Reads
  + Object Read (kb)
  + Object Writes
  + Object Write (kb)
  + Workload name
  + Workload description

The MDX Querying – Workload Comparison Summary, MDX Querying - Workload Comparison Detail, MDX Querying - Multiple Executions of a Specific Query, and DMV Session Info By Session reports all link to this report. This sample report also includes a link via the Object ID column to DMV Object Detail By Time that enables you to perform more analysis (the details of each report is described later in this section).

Figure 13 illustrates the execution of this report for the Sample Workloads for the most expensive processing session. Notice that you can view the various objects touched during this processing operation and the respective resources utilized during the processing of each object.







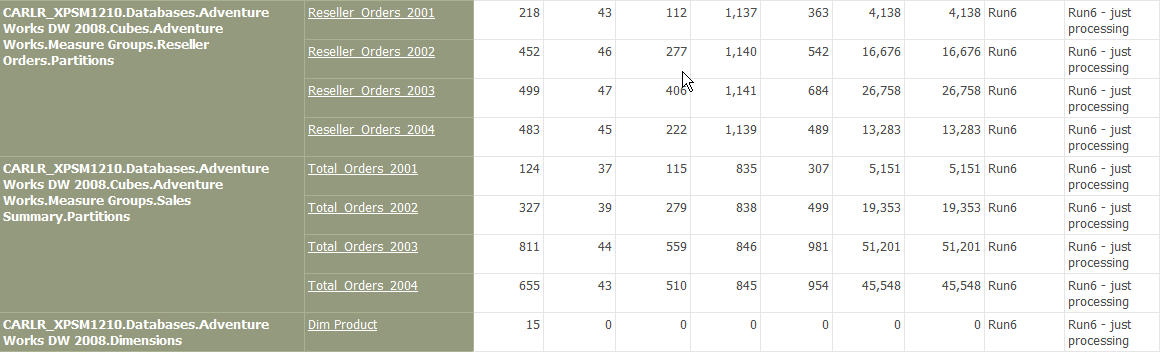
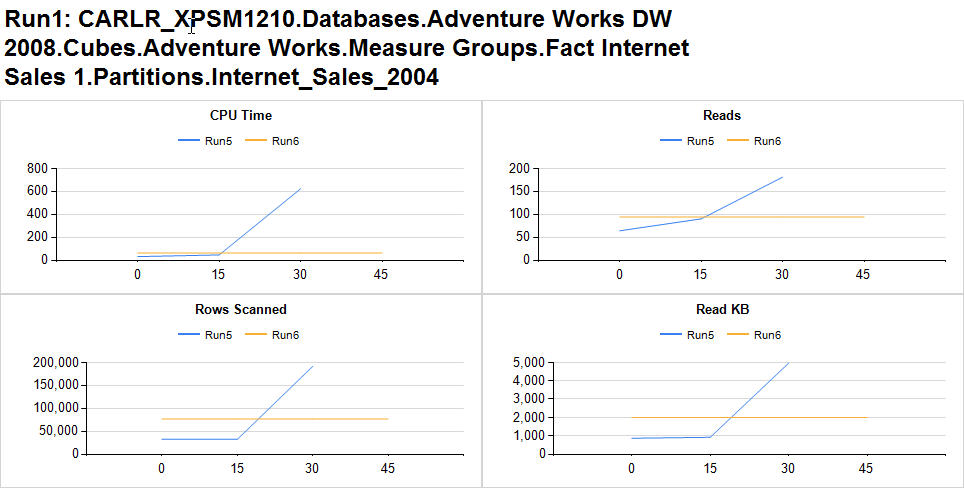


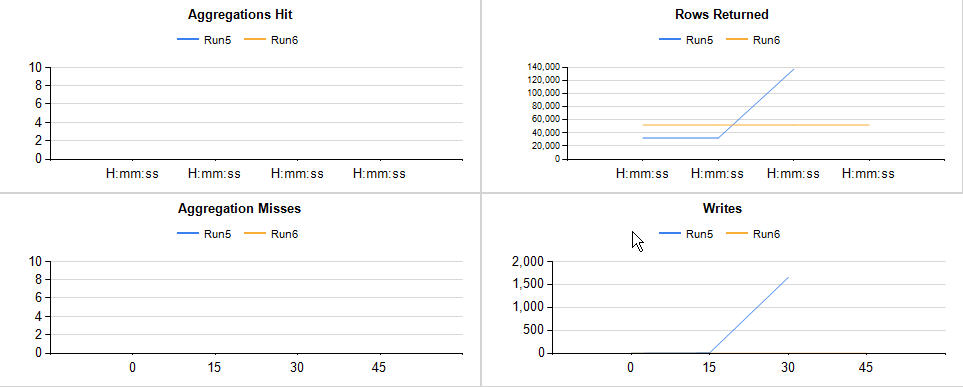
Figure 13: ObjectDetail\_PerSession

* **DMV Object Detail By Time** – This sample report analyzes Analysis Services DMV data and returns information about object activity for a particular object during the time period defined by a workload, and to compare that workload with other workloads. You can use this report to compare multiple workloads on an object by object basis. The report includes the following items for each workload:
  + CPU Time
  + Reads
  + Rows Scanned
  + Read KB
  + Aggregations Hit
  + Rows Returned
  + Aggregation Misses
  + Writes
  + Write KB
  + Shrinkable Memory
  + Non-Shrinkable Memory

The ObjectDetail\_PerSession report links to this report. This sample report also includes a link to DMV Object Detail By Time that enables you to perform more analysis (the details of each report is described later in this section). This report links to no other reports.

Figure 14 illustrates the execution of this report for the Sample Workloads for the Internet Sales 2004 partition for Runs 5 and 6. Notice that the time period covered by Run 5 uses more CPU time, has more reads, scans more rows, has more read KB, has more rows returned, has more writes, has more write KB, has more shrinkable memory, and has more non-shrinkable memory.





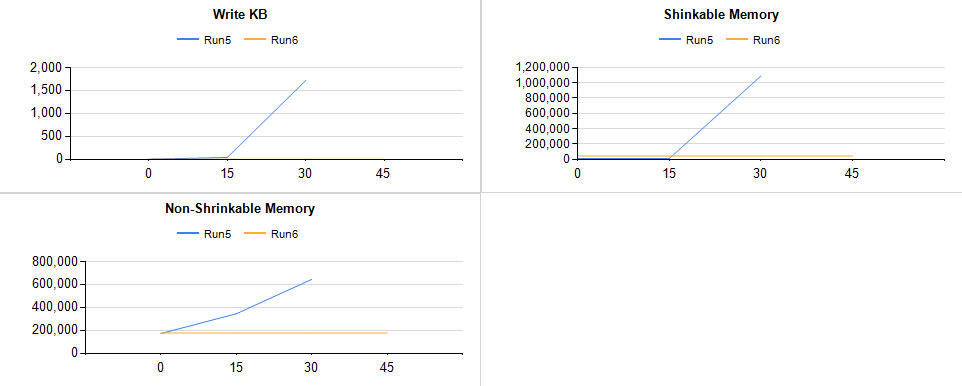


Figure 16: ObjectDetail\_PerSession

### Analyzing Processing

* **SSAS Processing – Major Objects** – This sample report analyzes Analysis Services trace data for the processing operations in one or more processing workloads, comparing the elapsed time for each major object (dimension or measure group object). This report enables you to compare the amount of time taken to process each major object in each workload, as well as view which objects are being processed simultaneous, versus objects that cannot be processed until other objects have completed their processing. This information will assist you in determining where to spend your time most profitably when tuning processing performance.

This sample report also includes links that enable you to perform more analysis (the details of each report is described later in this document):

* + Link by **Major Object** (dimension or measure group)– This is a link from a specified dimension or measure group object to the SSAS Processing – Specified Dimension or Cube Object report. This report is discussed later in this section.
  + **Link to Perf Monitor Data** – This is a link to the CountersDataPoints report, which enables you to view selected performance monitor counters for the time periods represented by the workloads. This report was discussed earlier in this section.
  + **Link to AS DMV Data by Time** – This is a link to the DMV Session Info By Time report, which enables you to view all DMV activity for the time periods represented by the workloads. This report was discussed earlier in this section.

**Important**: Hovering over the bar for a major object will cause the display of a tool tip containing the name of the object as well as the start and end time for the processing of that object. Clicking on that object passes those parameters to the SSAS Processing – Specified Dimension or Cube Object report.

Figure 17 illustrates the execution of the SSAS Processing – Major Objects report for Runs 5 and 6 of the Sample Workload. Notice that the elapsed time for each major object is about the same for each run.

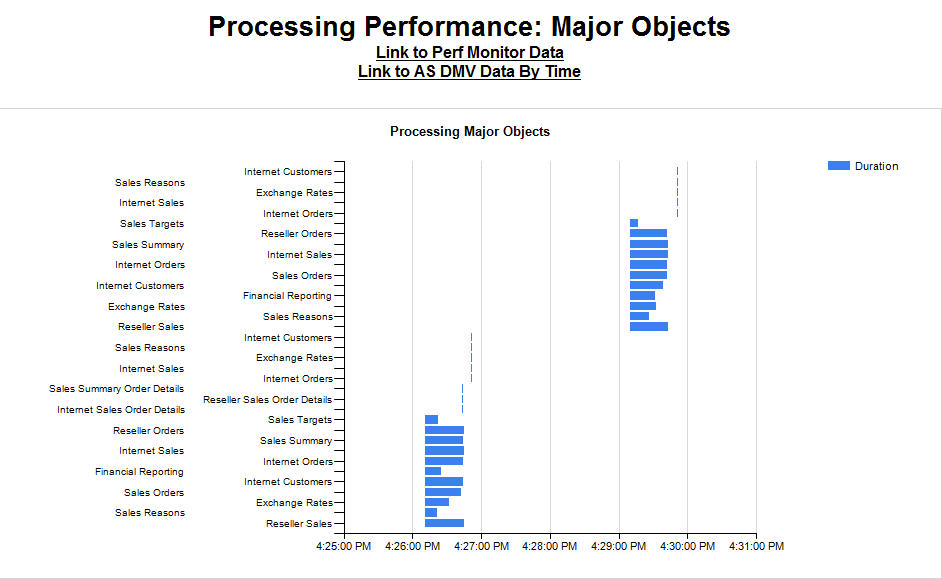


Figure 17: SSAS Processing – Major Objects report showing Runs 5 and 6

* **SSAS Processing – Specified Dimension or Cube Object** – This sample report analyzes Analysis Services trace data for the processing operations in a processing workload, comparing the elapsed time for each minor object (partitions for measure groups or attributes for dimensions). This report enables you to compare the amount of time taken to process each minor object that belongs to a major object, as well as view the parallel execution of multiple objects. This information will assist you in determining where to spend your time most profitably when tuning processing performance.

The SSAS Processing – Major Objects report links to this report. This sample report also includes links that enable you to perform more analysis (the details of each report is described later in this document):

* + Link by **Minor Object** (partition or attribute) – This is a link from a specified partition or attribute object to the SSAS Processing – Processing Details for a Specified Dimension or Cube Object report. This report is discussed later in this section.
  + **Link to Perf Monitor Data** – This is a link to the CountersDataPoints report, which enables you to view selected performance monitor counters for the time periods represented by the workloads. This report was discussed earlier in this section.
  + **Link to AS DMV Data by Time** – This is a link to the DMV Session Info By Time report, which enables you to view all DMV activity for the time periods represented by the workloads. This report was discussed earlier in this section.

**Important**: Hovering over the bar for a minor object will cause the display of a tool tip containing the name of the object as well as the start and end time for the processing of that object. Clicking on that object passes those parameters to the SSAS Processing – Processing Details for a Specified Dimension or Cube Object report.

Figure 18 illustrates the execution of the SSAS Processing – Specified Dimension or Cube Object report for the Reseller Sales measure group for Run 5 of the Sample Workload. Notice that the elapsed time for the 2004 and 2001 partitions are about the same, while the elapsed time for the 2002 and 2003 partitions are slightly less (although for this workload, we are only talking about a few seconds).

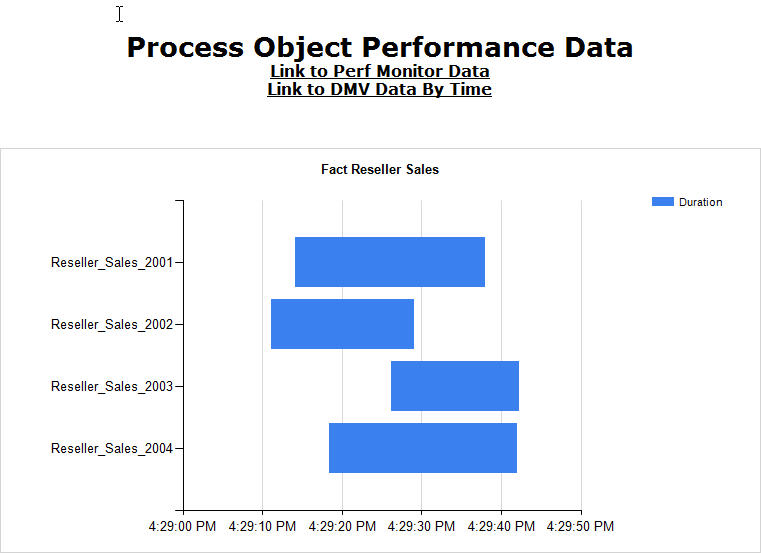


Figure 18: SSAS Processing – Specified Dimension or Cube Object report showing Reseller Sales partitions from Run 5

* **SSAS Processing – Processing Details for a Specified Dimension or Cube Object** – This sample report analyzes Analysis Services trace data for the processing operations in a processing workload, comparing the elapsed time for each processing task for processing the minor object. This report enables you to compare the amount of time taken by each processing task. This information will assist you in determining where to spend your time most profitably when tuning processing performance.

The SSAS Processing – Specified Dimension or Cube Object report links to this report.

**Important**: Hovering over the bar for a processing task will cause the display of a tool tip containing the name of the processing as well as the start and end time for the processing of that object.

Figure 19 illustrates the execution of the SSAS Processing – Processing Details for a Specified Dimension or Cube Object report for the Reseller\_Sales\_2004 partition for Run 5 of the Sample Workload. Notice that the vast majority of the time was spent building aggregations and indexes.

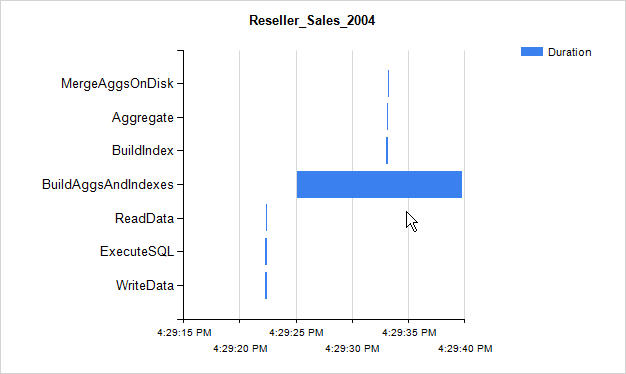


Figure 18: SSAS Processing – Specified Dimension or Cube Object report showing Reseller Sales partitions from Run 5

Appendix A: Working With the Files

**Important**: All files assume that the data collection database is called MDW. To use another database name, you will need to modify most of the files in this solution.

## FOLDER: MonitoringSSAS

This folder describes the files in the MonitoringSSAS folder that are used to configure this solution. These files are arranged in alphabetical order

1. **Configuring\_MDWDataCollectionServer.cmd** – This batch file calls the SetVariables.cmd batch file to set environment variables and then calls the scripts and Integration Services packages in the Configuring\_Data\_Collection\_Server folder to configure the MDWDataCollectionServer.
2. **Configuring\_SSASServer.cmd** – This batch file calls the SetVariables.cmd batch file to set environment variables and then calls the scripts in the Configuring\_SSAS\_Server folder to configure the SSASServer.
3. **IdentifyingAndResolvingMDXBottlenecksSSAS.doc** – a white paper explaining how to interpret SQL Profiler traces.
4. **Reset\_Collected\_Data.cmd** – This batch file calls the SetVariables.cmd batch file to set environment variables and then calls the scripts in the Resetting\_Data\_Collection folder to reset the solution by truncating data collection tables and resetting identity columns on the MDWDataCollectionServer.
5. **SetVariables.cmd** – This batch file sets environment variables for your environment.
6. **SSASPerfGuide2008.docx** – a white paper explaining how to tune Analysis Services 2008 for performance.
7. **Start\_Collection.cmd** – This batch file calls the SetVariables.cmd batch file to set environment variables and then calls the scripts in the Starting\_Data\_Collection folder to start data collection. Customize this file if you wish to collect from only some of the data sources in this data collection solution.
8. **Stop\_Collection.cmd** – This batch file calls the SetVariables.cmd batch file to set environment variables and then calls the scripts in the Stopping\_Data\_Collection folder to stop data collection.

## FOLDER: Configuring\_Data\_Collection\_Server

This folder describes the files in the Configuring\_Data\_Collection\_Server folder that are used to configure the MDWDataCollectionServer for data collection and data analysis. These files are arranged in alphabetical order.

1. **Add\_schemas\_db\_mdw.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the following schemas in the MDW database: MDW, SSAS, Stage, and Custom\_Snapshots.
2. **AS\_TraceData.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the AS\_TraceData table in the MDW database. This SQL Server table stores data from each trace file loaded from the file system before any processing of the loaded trace data occurs. After each processing of loaded trace data, all data related to a completed query, command or processing operation is deleted from this file. Query data related to partially completed queries is retained in this table and combined with the data from the subsequent run of the Integration Services package. The trace data loaded into this file is ordered by SessionID and EndTime as part of each load of this table by the Integration Services package.
3. **Create\_DB\_MDW\_Control.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script drops, if necessary, and then creates the mdw\_control database for reporting metadata, and creates the required schemas (mdw and ssas).
4. **Drop\_and\_create\_db\_mdw.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script drops, if necessary, and then creates the MDW database.
5. **Event\_class.dtsx** – This Integration Services package is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This package loads the EventClass.csv data into the EventClass table in the MDW database.
6. **EventClass**.csv – This CSV file contains user-friendly information regarding each event class code.
7. **Event\_subclass.dtsx** – This Integration Services package is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This package loads the EventSubclass.csv data into the EventClass table in the MDW database.
8. **EventSubclass.csv** – This CSV file contains user-friendly information regarding each event subclass code.
9. **Get\_command\_objects\_by\_id.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns DMV information about command object activity related to a specific session. This
10. **Get\_counter\_data\_points\_by\_config.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns counter data points to a report from the counters specified via the set\_counter\_config\_by\_list stored procedure.
11. **Get\_memory\_usage\_from\_list.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns DMV memory information related to objects based on time.
12. **Get\_object\_activity\_from\_list.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns DMV processor and I/O information related to object activity based on time. This file does not require modification.
13. **Get\_processing\_major\_objects.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns information about the processing of major objects (dimensions and measure groups) based on time.
14. **Get\_processing\_minor\_objects.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns information about the processing of a specified dimension attribute or partition.
15. **Get\_processing\_minor\_objects\_detail.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns information about the processing of a specified dimension or measure group.
16. **Get\_queries\_full.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns performance information about MDX query performance based on time of execution, duration, CPU time, and number of executions.
17. **Get\_query\_execution.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns performance information about MDX query performance of multiple executions of a specified query based on time of execution, duration, CPU time, and number of executions.
18. **Get\_query\_execution\_subcube.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns storage engine performance information about the execution of a specified query.
19. **Get\_query\_text.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns the MDX query text of a specified query.
20. **Get\_session\_dmv\_info\_by\_session.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns DMV information related to a specified session.
21. **Get\_session\_dmv\_info\_by\_time.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that returns DMV information related to connections, sessions, and commands based on time. This file does not require modification.
22. **Load\_perfmon\_counter\_config\_table.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database for loading valid performance monitor counters for an Analysis Services instance for a specified period of time into the perfmon\_counter\_config table in the MDW database.
23. **Load\_SSAS\_trace\_data.dtsConfig** – This Integration Services configuration file is used by the Load\_SSAS\_trace\_data.dtsx Integration Services package called by the SSAS\_Trace job. This configuration file specifies the location of the trace files.
24. **Load\_SSAS\_trace\_data.dtsx** – This Integration Services package is executed by the SSAS\_Trace job and loads the trace files into the MDW database. The Load\_SSAS\_trace\_data.dtsConfig configuration file specifies the location of the trace files.
25. **Read\_SSAS\_DMV.dtsConfig** – This Integration Services configuration file is used by the Read\_SSAS\_DMV.dtsx Integration Services package called by the SSAS\_DMVs job. This configuration file specifies the location of the trace files.
26. **Read\_SSAS\_DMV.dtsx** – This Integration Services package is executed by the SSAS\_DMVs job and loads DMV data into the MDW database. The **Read\_SSAS\_DMV.dtsConfig** configuration file specifies the SSASServer that is queried.
27. **Set\_counter\_config\_by\_list.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW\_Control database that takes an input list to configure the list of performance monitor counters to be returned to a report.
28. **SSAS\_Command\_Objects.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAS\_Command\_Objects table in the MDW database that stores Analysis Services DMV information related to command objects related to each session.
29. **SSAS\_Commands.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAS\_Commands table in the MDW database that stores Analysis Services DMV information related to completed commands.
30. **SSAS\_Connections.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAS\_Connections table in the MDW database that stores Analysis Services DMV information related to completed connections.
31. **SSAS\_DMVs.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAS\_DMVs disabled job in a specified SQL Server instance to collect and load AS DMV data from a specified Analysis Services instance into trace tables.
32. **SSAS\_Memory\_Usage.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAS\_Memory\_Usage table in the MDW database that stores Analysis Services DMV information related to memory usage by objects over time.
33. **SSAS\_Object\_Activity.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAS\_Object\_Activity table in the MDW database that stores Analysis Services DMV information related to processor and I/O activity by objects over time.
34. **SSAS\_Sessions.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAS\_Sessions table in the MDW database that stores Analysis Services DMV information related to completed sessions.
35. **SSAS\_Trace.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAS\_Trace disabled job in a specified SQL Server instance to load trace data into trace tables from a specified location in the file.
36. **SSASAggregation.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASAggregation table in the MDW database that stores data related to each unique aggregation in each measure group in each unique cube in each unique database in each instance of Analysis Services from which trace data containing MDX query information is received.
37. **SSASBatch.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASBatch table in the MDW database that generates a BatchID value and records a time stamp for each load of trace files into the trace data tables. This BatchID is recorded in each fact table, which enables you to back out a specific data load if necessary.
38. **SSASCleanBatch.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW database deletes all rows in theSSASWorkTracetable table that relate to completed queries and remove any row identification metadata for incomplete queries that were written by the SSASResolveLinks stored procedure. It also deletes data from incomplete queries related to any previous batch.

**Note**: There is an assumption being made by the SSASCleanBatch stored procedure that incomplete queries will not remain incomplete across more than one batch. This may need to be adjusted depending on your frequency of trace file loads and the typical length of your MDX queries.

1. **SSASCube.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASCube table in the MDW database that stores data related to each unique cube in each unique database in each instance of Analysis Services from which trace data containing MDX query information is received.
2. **SSASDatabase.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASDatabase table in the MDW database that stores data related to each unique database in each instance of Analysis Services from which trace data containing MDX query information is received.
3. **SSASDMVCommand** **.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASDMVCommand table in the MDW database that is a staging table for recording Analysis Services DMV data related to current commands.
4. **SSASDMVConnection.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASDMVConnection table in the MDW database that is a staging table for recording Analysis Services DMV data related to current connections.
5. **SSASDMVSession.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASDMVSession table in the MDW database that is a staging table for recording Analysis Services DMV data related to current sessions.
6. **SSASEventClass.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASEventClass table in the MDW database that stores user-friendly information regarding each Analysis Services trace event class.
7. **SSASEventSubclass.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASEventSubclass table in the MDW database that stores user-friendly information regarding each Analysis Services event subclass.
8. **SSASInstance.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASInstance table in the MDW database that stores data related to each unique instance of Analysis Services from which trace data containing MDX query information is received.
9. **SSASLoadFromTraceTable.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW database that loads MDX query data from the AS\_TraceData table into the SSASWorkTracetable table and Analysis Services processing data into the SSASProcessingArchive table. As part of this process, it calculates and stores in a new column the DATETIMEOFFSET value for the start and end times for each record – these values are stored in the trace file using local time only. Specifically, the [ssas].[SSASLoadFromTraceTable] stored procedure that is created by this script file uses the DATEPART and SYSDATETIMEOFFSET functions to determine the offset to be used when loading trace data stored using local time into the trace tables.

**Important:** The offset is calculated once for each execution of this stored procedure. If you are loading trace data that for a time period that includes a change in the datetimeoffset due to a time change, such as a change to daylight savings time, you will need to manually adjust your data after loading or modify this stored procedure to account for such changes.

**Note**: You may wish to eliminate DMV query events from your trace files – you can eliminate them either in this stored procedure or in the **load\_SSAS\_trace\_data.dtsx** Integration Services package.

1. **SSASMeasureGroup.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASMeasureGroup table in the MDW database that stores data related to each unique measure group in each unique cube in each unique database in each instance of Analysis Services from which trace data containing MDX query information is received.
2. **SSASPartition.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSAPartition table in the MDW database that stores data related to each unique partition in each measure group in each unique cube in each unique database in each instance of Analysis Services from which trace data containing MDX query information is received.
3. **SSASProcessingArchive.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASProcessingArchive table in the MDW database that stores trace data related to Analysis Services processing data.
4. **SSASQuery.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASQuery table in the MDW database that stores each unique query. Unique queries are identified using string comparison.
5. **SSASQueryExecution.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASQueryExecution table in the MDW database that stores data related to each execution of an MDX query.
6. **SSASQueryExecutionEvents.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASQueryExecutionEvents table in the MDW database that stores performance data related to each execution of an MDX query. The table contains data aggregate data recorded in the profiler trace as well as data aggregated from the SSASQuerySubCubeEvents and SSASQueryPartitionEvents tables. The data includes the total duration, the storage engine duration, the formula engine duration, the number of cached and noncached subcubes, the number of partitions, the number of aggregations, the number of measure groups, and the start and end times for each query.
7. **SSASQueryPartitionEvents.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASQueryPartitionEvents table in the MDW database that stores performance data related to each partition queried by the storage engine for each execution of an MDX query. The data includes the aggregation used (if any) and the duration of the partition event.
8. **SSASQuerySubCubeEvents.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASQuerySubCubeEvents table in the MDW database that stores performance data from partition events correlated to subcube events for an execution of an MDX query. Subcube events that are not associated with partition events are ignored in this table. By correlating subcube events with partition events, we are able to calculate the amount of time spent by the storage engine in resolving an MDX query. The data includes whether the subcube request was resolved from cache, the storage engine duration, and the number of aggregations (if any).
9. **SSASRefreshDimensions.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW database that updates the dimension tables with new data. The tables updated are:
   * SSASUser
   * SSASInstance
   * SSASDatabase
   * SSASCube
   * SSASMeasureGroup
   * SSASSubCube
   * SSASPartition
   * SSASAggregation
   * SSASQuery
   * SSASQueryExecution
10. **SSASRefreshFacts.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW database that adds new facts to the fact tables. The tables to which new rows are added are:
    * SSASQueryPartitionEvents
    * SSASQuerySubCubeEvents
    * SSASQueryExecutionEvents
11. **SSASResolveLinks.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates a stored procedure in the MDW database that identifies the rows in theSSASWorkTracetable table associated with each query, the rows in each query that are subcube events, and the subcube event row to which each partition event row is related. To accomplish this, the data in the **SSASWorkTracetable** table must be properly sorted by SessionID and EndTime.
12. **SSASSubCube.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASSubCube table in the MDW database that stores data related to each subcube for each MDX query in the trace data.
13. **SSASUser.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASUser table in the MDW database that stores each unique user.
14. **SSASWorkTable\_Transform.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASWorkTrace table in the MDW database that is a view of all of the columns in the SSASWorkTracetable table and the stored procedures in this solution query through this view rather than against the underlying table directly.
15. **SSASWorkTraceTable.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the SSASWorkTraceTable table in the MDW database that is a work table in which the MDX query data is stored during the processing of the MDX query information from each load of trace files. This file is truncated after each load completes.
16. **Workload\_Runs.sql** – This Transact-SQL script is executed by the Configuring\_MDWDataCollectionServer.cmd batch file. This script creates the Workload\_Runs table in the MDW\_Control database that stores metadata for reporting purposes.

## FOLDER: Configuring\_SSAS\_Server

This folder describes the files in the Configuring\_SSAS\_Server folder that are used to configure the SSAS Server for data monitoring. These files are arranged in alphabetical order.

1. **Purge\_Core\_MDW.sql** – This Transact-SQL script purges any information collected by the System Collection Sets before you stopped them. These collection sets are started automatically when you initially configure Management Data Warehouse on SSASServer.
2. **SSAS\_Monitoring\_Core.sql** – This Transact-SQL script drops, if necessary, and creates the ssas\_monitoring\_core custom collection set. This file contains the list of Performance Monitor counters monitored. Modify this list to customize the counters for which data is collected.
3. **Stop\_System\_Collection\_Sets.sql** – This Transact-SQL script stops the Management Data Warehouse System Collection Sets. These collection sets are started automatically when you initially configure Management Data Warehouse on SSASServer.

## FOLDER: Collecting\_SSAS\_Perf\_Mon\_Data\_Verbose

This folder contains the SSAS\_Monitoring\_Verbose.sql Transact-SQL script that you can use for creating a custom SSAS\_Monitoring\_Verbose data collection set in Management Data Warehouse. This collector will collect all remaining Analysis Services performance monitor counters not collected by the SSAS\_Monitoring\_Core collector. This collector is not currently used in this data collection solution.

## FOLDER: Collecting\_SSAS\_DMV\_Data\_TSQL\_Collector

This folder contains the files that you may use to create a custom collection set using the T-SQL collector to collect DMV data. The files in this folder are discussed in this section, including a description of what the file does and how you may need to modify it. These files are not currently used in this data collection solution.

1. **AddLinkedServer.sql** – This Transact-SQL script uses the sp\_addlinkedserver system stored procedure to define a linked server from the MDWDataCollectionServer to the SSAS Server. Use the information in the following table to customize this file.

|  |  |  |
| --- | --- | --- |
| **Value Type** | **Default Value** | **New Value** |
| @datasrc | SSASServer | SSAS Server instance name |
| @catalog | YourDatabaseNameHere | Analysis Services database name |

1. **DMV\_Views\_OpenQuery.sql** – This Transact-SQL script creates SQL Server views containing DMV queries issued via OPENQUERY against your SSAS Server. This file does not require modification.
2. **SSAS\_DMVs.sql** – This Transact-SQL script creates the SSAS\_TSQL\_DMVs custom T-SQL collector in Management Data Warehouse. Use the information in the following table to customize this file.

|  |  |  |
| --- | --- | --- |
| **Value Type** | **Default Value** | **New Value** |
| @days\_until\_expiration | 14 | Your choice |
| PerformanceCounters Objects | Selected objects | Your choice |
| @frequency | 60 | Your choice |

## FOLDER: Starting\_Data\_Collection

This folder describes the files used to start data collection. These files are arranged in alphabetical order.

1. **SSASMonitoringTrace.tdf** – The SQL Profiler template for Analysis Services from which the server-side trace was defined.
2. **Start\_Collecting\_SSAS\_Perf\_Mon\_Data.sql** – This Transact SQL script starts the SSAS\_Monitoring\_Core custom collection set on the specified SSASServer.
3. **Start\_DMV\_Collection.sql** – This Transact-SQL file script starts the SSAS\_DMVs job on the specified MDWDataCollectionServer to query a specified SSASServer.
4. **Start\_SSAS\_Server\_Trace.xmla** – This xmla file starts a server-side trace on the specified SSAS Server.
5. **Start\_Trace\_Upload.sql** – This Transact-SQL script starts the SSAS\_Trace job on the MDWDataCollectionServer to upload trace file data from a specified file location on the specified MDWDataCollectionServer.

## FOLDER: Stopping\_Data\_Collection

This folder describes the files used to stop data collection. These files are arranged in alphabetical order.

1. **Execute\_MDW\_Load\_Perfmon\_Counter\_Config\_Table.sql** – This Transact-SQL script calls the load\_perfmon\_counter\_config\_table stored procedure to populate the perfmon\_counter\_config table in the MDW\_Control database with the Performance Monitor counters for which you have collected data for your Analysis Services instance. This table is created and populated each time you run this stored procedure.
2. **Stop \_Collecting\_SSAS\_Perf\_Mon\_Data.sql** – This Transact-SQL script stops the SSAS\_Monitoring\_Core on the specified SSASServer.
3. **Stop \_DMV\_Collection.sql** – This Transact-SQL file script stops the SSAS\_DMVs job on the specified MDWDataCollectionServer.
4. **Stop\_SSAS\_Server\_Trace.xmla** – This xmla file stops the server-side trace on the specified SSAS Server.
5. **Stop \_Trace\_Upload.sql** – This Transact-SQL script stops the SSAS\_Trace job on the specified MDWDataCollectionServer.
6. **Upload\_Trace\_Data.sql** – This Transact-SQL script uploads collected trace data after the server-side trace is stopped to ensure that all collected trace data is loaded in the MDW database in the specified MDWDataCollectionServer as part of stopping the data collection.

## FOLDER: Resetting\_Data\_Collection

This folder describes the files used to reset data collection by truncating data collection tables and resetting identity columns on the MDWDataCollectionServer. The files in this folder are discussed in this section, including a description of what the file does and how you may need to modify it.

1. **Clear\_Data\_SSAS\_monitoring\_tables.sql** – This Transact-SQL file truncates and resets the Analysis Services trace data tables on the MDWDataCollectionServer.
2. **Purge\_Core\_MDW.sql** – This Transact-SQL file purges the data in Management Data Warehouse related to the specified SSASServer.
3. **Purge\_SSAS\_DMV.sql** – This Transact-SQL file truncates and resets the Analysis Services DMV tables on the MDWDataCollectionServer.

## FOLDER: Populating\_OLAP\_Query\_Log

This folder describes the files that you can use to populate the OLAP Query Log with data collected by this data collection solution.

1. **Create\_Table\_SSAS\_OLAPQueryLog.sql** – This Transact-SQL script creates the OLAPQueryLog table in the MDW\_Control database on the MDWDataCollectionServer.
2. **Populate\_OLAP\_Query\_Log.sql** – This Transact-SQL script creates the Populate\_OLAP\_Query\_Log\_Table stored procedure in the MDW\_Control database on the MDWDataCollectionServer. You can use this stored procedure to load the OLAPQueryLog table with selected query log data about your problem queries stored in the MDW database and then create aggregations based on the subcube query vectors recorded in this table. This stored procedure requires the following parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Value** |
| @mdw\_database\_name | The name of the MDW database | MDW |
| @InstanceName | The (an) SSAS Server instance name | Your choice |
| @DatabaseName | The name of the database on the SSAS Server | Your choice |
| @DurationThreshold | Return only information relating to queries exceeding this threshold | Your choice |
| @CPUTimeThreshold | Return only information relating to queries exceeding this threshold | Your choice |
| @begin\_time | Return only information relating to queries after this time | Your choice |
| @end\_time | Return only information relating to queries before this time | Your choice |

**Note**: Specify begin times after any structural change to your database. Returning query information from a time period prior to a structural change may return invalid subcube query vectors.

## FOLDER: Analyzing\_SSAS\_Perf\_Data\_on\_Monitoring\_Server

This folder contains two folders, Scripts and SSAS\_Reports, and contains the SSAS\_Reports.sln file. This solution file is the solution file for the reports and data sources in the SSAS\_Reports folder.

### FOLDER: Scripts

The files in this folder are discussed in this section, including a description of what the file does and how you may need to modify it.

1. **Insert.sql** – This Transact-SQL script inserts rows of data into the workload\_runs table. Modify the data in this file for your environment.

### FOLDER: SSAS\_Reports

This folder contains 12 sample reports and the data source for the sample reports. The files in this folder were discussed in detail earlier in this document in the Using the Sample Reports to Query the Collected Data section of this document. The files in this folder are:

1. MDW\_Control.rds
2. CountersDataPoints.rdl
3. DMV\_ObjectDetail\_PerSession
4. DMV\_ObjectDetail\_PerTimeRange\_Select\_Object
5. DMV\_Session\_Info\_By\_Session
6. DMV\_Session\_Info\_By\_Time.rdl
7. MDX Querying - Details of an Execution of an Individual Query.rdl
8. MDX Querying - Multiple Executions of a Specific Query.rdl
9. MDX Querying - Workload Comparison Detail.rdl
10. MDX Querying - Workload Comparison Summary.rdl
11. SSAS Processing - Specified Dimension or Cube Object.rdl
12. SSAS Processing - Processing Details for a Specified Dimension or Cube Object.rdl
13. SSAS Processing - Major Objects.rdl

### FOLDER: DataForPaper

This folder contains the following items used to produce this paper:

* + Backup of the MDW database
  + Backup of the MDW\_Control database
  + ASMCD scripts used to execute queries as well as the queries themselves
  + Microsoft Excel spreadsheet containing output from the Sample Workload