Oracle® Essbase

Configuration Reference for Oracle Essbase

F17648-02
September 2020

Oracle Essbase Configuration Reference for Oracle Essbase,

F17648-02

Primary Author: Essbase Information Development Team

Contents

Accessibility and Support

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1

Configuration Reference Overview

You can use the Oracle Essbase configuration settings to customize the functionality of Essbase.

This reference describes available configuration options for Essbase. This reference is intended for advanced users who need detailed information and examples.

This document provides examples based mostly on the Sample Basic cube, provided with Essbase as a template you can build into a cube. The Sample application, as well as more samples you can build, are available in the Applications > Demo Samples section of the gallery. The gallery is available in the Files section of Essbase. See Explore the Gallery Templates.

To use this document, you need the following:

- A working knowledge of the operating system.
- An understanding of Essbase concepts and features.
- An understanding of the typical database administration requirements and tasks, including calculation, querying, security, and maintenance.

Set Application-Level Configuration Properties

If you have the Service Administrator role, or the Power User role for applications that you created, you can customize Oracle Essbase using application-level configuration properties. Application-level configuration properties apply to all cubes in the application.

One way to specify configuration properties of an application is to do it prior to building the application and cube, using the application workbook. To see an example, go to Files in the Essbase web interface, and download the application workbook `Sample_Basic.xlsx`. It is located in the gallery, in the Demo Samples section (under Block Storage). In this application workbook, go to the Cube.Settings worksheet. Under Application Configuration, you can see that the DATACACHESIZE property is set to 3M, and the INDEXCACHESIZE property is set to 1M.

The following steps tell you how to configure an application that is already deployed, by adding properties and their corresponding values in the Essbase web interface.

1. On the Applications page, select the application you want to configure.
2. From the **Actions** menu to the right of the application, click **Inspect**, then click **Configuration.**
3. To add a property, click +. Select a property from the list. When done adding properties, close the list window.
4. To change a property's value, double-click a property row and edit its value.
5. When you're finished making changes, click **Apply.**

   The configuration changes will take effect next time the application restarts.

For syntax and information about each of the application configuration properties you can use, see Config Settings List.

Oracle does not recommend that you modify `essbase.cfg` on the Essbase file system. This configuration is automatically set.

Set Provider Services Configuration Properties

If you have the Service Administrator role, you can customize network-related settings for Oracle Essbase using the Provider Services configuration properties.

To set the values for Provider Services configuration properties,

1. Log in to the Essbase web interface as a Service Administrator.
2. Click **Console**.
3. In the Console, click **Configuration**.
4. On the Provider Services tab, click **Add** to add a new property and set its value. If the property you want to configure is already listed, double-click the **Value** field to edit the value.
5. When you are finished editing properties, click **Save**.

2 Application Configuration Settings

You can use a variety of configuration settings to customize the behavior of your applications.

- Config Settings List
- Configuration Settings Categorical List
- Aggregate Storage and Block Storage Settings Comparison

To change application configuration properties, see Set Application-Level Configuration Properties

Config Settings List

- AGENTTHREADS
- ASODEFAULTCACHESIZE
- ASODYNAMICAGGINBSO
- ASODYNAMICAGGINBSOFOLDERPATH
- AUDITTRAIL
- AUTOMERGE
- AUTOMERGEMAXSLICENUMBER
- CALCCACHE
- CALCCACHEDEFAULT
- CALCCACHEHIGH
- CALCCACHELOW
- CALCLIMITFORMULARECURSION
- CALCPARALLEL
- CALCTASKDIMS
- CALCTRACE
- CUSTOMCALCANDALLOCTHRUINSERT
- DATACACHESIZE
- DEFAULTVIEWBUILD
- DEFAULTVIEWBUILDSIZE
- DLSINGLETHREADPERSTAGE
- DLTHREADSPREPARE
- DLTHREADSWRITE
- DYNCALCCACHEMAXSIZE

- ENABLERTSVLOGGING
- FORCEALLDENSECALCON2PASSACCOUNTS
- FORCESHUTDOWNINTERVAL
- GRIDEXPANSION
- GRIDEXPANSIONMESSAGES
- GRIDSUPPRESSINVALID
- HYBRIDBSOINCALCSCRIPT
- IDLETIMEMERGE
- IGNORECONSTANTS
- INDEXCACHESIZE
- LONGCALCTIMETHRESHOLD
- LONGQUERYTIMETHRESHOLD
- LONGREQTIMETHRESHOLD
- MAXFORMULACACHESIZE
- MAXLOGINS
- MAXNUMBEROFACTIVEDB
- MAX_REQUEST_GRID_SIZE
- MAX_RESPONSE_GRID_SIZE
- MDXINSERTBUFFERAGGMETHOD
- MDXINSERTREQUESTTIMEOUT
- MDXQRYGOVCOUNT
- MULTIPLEBITMAPMEMCHECK
- NUMBLOCKSTOEXTEND
- PARCALCMULTIPLEBITMAPMEMOPT
- QUERYRESULTLIMIT
- QRYGOVEXECBLK
- QRYGOVEXECTIME
- QUERYTRACE
- QUERYTRACETHRESHOLD
- RENEGADELOG
- RESTRUCTURETHREADS
- RTDEPCALCOPTIMIZE
- SERVERTHREADS
- SSANCESTORONTOP
- SSMEMBERIDPROCESSING
- SSOPTIMIZEDGRIDPROCESSING
- SSPROCROWLIMIT

AGENTTHREADS

Specifies the maximum number of threads that Essbase can spawn for operations such as logging in and out of Essbase Server and starting and stopping an application.

Syntax

```
AGENTTHREADS n
```

n—Specifies the number of threads that Essbase can spawn, where *n* can be 5 to 500, inclusive.

The default value is 5.

Notes

- While the actual maximum value you can set is 500, the maximum number of threads an operating system can handle might be much lower. Before specifying a value greater than the default value, check with your system administrator, as higher values can significantly consume system resources.
- If you specify a number that is less than 5, over the maximum, or a decimal value, Essbase overrides the value with a closely approximate value of its own.
- One thread is required for each initial connection to an application and database.
- The AGENTTHREADS configuration setting does not apply to Essbase Java Agent, which uses the WebLogic Server thread pool configuration for the total number of threads that can be spawned at the server and domain levels. This total thread count is limited to 500 and is specified in the `config.xml` file. If the value of AGENTTHREADS is less than the value of the WebLogic Server total thread count, Essbase uses the value specified in AGENTTHREADS; if the value of AGENTTHREADS is more than the WebLogic Server total thread count, Essbase uses the value specified in the `config.xml` file.

Example

```
AGENTTHREADS 100
```

Sets the maximum number of threads that Essbase can spawn to 100, assuming that the total thread count specified in the `config.xml` file is 100 or more.

ASODEFAULTCACHESIZE

Sets the default size for the aggregate storage cache associated with aggregate storage databases. The aggregate storage cache grows dynamically until it reaches this limit.

This setting applies only to aggregate storage databases.

Syntax

```
ASODEFAULTCACHESIZE [appname] n
```

- *appname*—Optional. Specifies the application to which the setting applies. If omitted, the setting applies to all new applications.
- *n*—An integer value indicating size in megabytes.

Description

ASODEFAULTCACHESIZE specifies, in megabytes, the size of the aggregate storage cache for aggregate storage databases.

The aggregate storage cache facilitates memory usage during data loads, aggregations, and retrievals. When an aggregate storage application is started, Essbase allocates a small area in memory as the aggregate storage cache for the application. As needed, Essbase increases the cache size incrementally until the maximum cache size specified for the application is reached or until the operating system denies additional allocations.

Example

```
ASODEFAULTCACHESIZE 200
```

Sets the aggregate storage cache size of all newly created or migrated aggregate storage databases as 200 megabytes.

ASODYNAMICAGGINBSO

Controls whether block storage databases use hybrid mode for queries. Hybrid mode for block storage databases means that wherever possible, block storage queries execute with efficiency similar to that of aggregate storage databases.

Hybrid mode is enabled by default for queries (this configuration setting is implicitly set to FULL). To enable hybrid mode for batch calculations, you must enable HYBRIDBSOINCALCSCRIPT in the application configuration.

This setting applies only to block storage databases.

Syntax

```
ASODYNAMICAGGINBSO [appname [dbname]] NONE | PARTIAL | FULL | ONLY
```

- *appname*—Optional. Specifies the application for which hybrid query mode is used in aggregations.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which hybrid mode is used in aggregations.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- NONE—Disable hybrid mode in block storage databases. This is the default.
- PARTIAL—Turn on hybrid mode only for simple outline aggregations based on the consolidation operators `+`, `-`, and `~`, but excluding the operators `*`, `/`, and `%`. Leave formulas to be calculated in block storage mode.
- FULL—Turn on hybrid mode for simple aggregations and formula calculations. If enabled, hybrid mode is in effect for member formulas using any of the supported functions. For a list of supported and unsupported functions, see Functions Supported in Hybrid Aggregation Mode.
- ONLY—Same as FULL, but if a query is not supported in hybrid mode, return an error instead of defaulting to block storage execution. This can be useful for testing purposes while you are migrating a database from block storage execution to hybrid mode.

Example

```
ASODYNAMICAGGINBSO Sample PARTIAL
```

See Also

Adopt Hybrid Mode for Fast Analytic Processing.

MaxL statements alter application **set cache_size** and query application **get cache_size**, for managing the size of block-storage application cache.

ASODYNAMICAGGINBSOFOLDERPATH

Changes the location specification for hybrid aggregation mode directories.

This setting applies only to block storage databases.

When a block storage database uses hybrid aggregation mode, the following subdirectories are created under the application directory:

- `default`
- `log`
- `metadata`
- `temp`

These subdirectories are similar to those found in aggregate storage application directories. When the application stops, the directories are removed, and when the application restarts, they are replaced.

Syntax

```
ASODYNAMICAGGINBSOFOLDERPATH [appname] path_to_directory
```

- *appname*—Optional application specification.

  If you do not specify an application, the setting applies to all applications and databases on Essbase Server.
- *path_to_directory*—Path to the new directory after you have moved it.

Example

```
ASODYNAMICAGGINBSOFOLDERPATH Sample \\machine-name\directory
```

AUDITTRAIL

Use an audit trail to track changes to cube data, including Smart View updates, changes to Linked Reporting Objects (LROs), adding notes, attaching files, and referencing URLs.

This setting is applicable only for block storage cubes.

You can view the audit log in Smart View, and also in the Essbase web interface.

Syntax

```
AUDITTRAIL DATA
```

Example

```
AUDITTRAIL DATA
```

AUTOMERGE

Specifies whether incremental data slices are automatically merged during a data load to an aggregate storage database.

This setting applies only to aggregate storage databases.

Syntax

```
AUTOMERGE ALWAYS | NEVER | SELECTIVE
```

- ALWAYS—Specifies to automatically merge incremental data slices during a data load to an aggregate storage database. By default, merges are executed once for every four consecutive incremental data slices. If, however, the AUTOMERGEMAXSLICENUMBER configuration setting is used, the auto-merge

process is activated when the AUTOMERGEMAXSLICENUMBER value is exceeded.

The size of the incremental data slices is not a factor in selecting which ones are merged.

The default value is ALWAYS.

- NEVER—Specifies to never automatically merge incremental data slices during a data load to an aggregate storage database.

  To manually merge incremental data slices, use the **alter database** MaxL statement with the **merge** grammar.

- SELECTIVE—Specifies to activate the incremental data slice auto-merge process when the number of incremental data slices specified in the AUTOMERGEMAXSLICENUMBER configuration setting is exceeded. If the number of incremental data slices in the data load does not exceed the value of AUTOMERGEMAXSLICENUMBER, the auto-merge process is not activated.

Example

```
AUTOMERGE SELECTIVE
```

Specifies that the value of the AUTOMERGEMAXSLICENUMBER configuration setting determines whether the process of automatically merging incremental data slices is activated.

See Also

AUTOMERGEMAXSLICENUMBER

AUTOMERGEMAXSLICENUMBER

Specifies the maximum number of incremental data slices that can exist in a data load without activating the process of automatically merging incremental data slices. When the value of AUTOMERGEMAXSLICENUMBER is exceeded, the auto-merge process is activated.

> **Note:**
>
> To use the AUTOMERGEMAXSLICENUMBER configuration setting, the AUTOMERGE configuration setting must be set to SELECTIVE or ALWAYS.

This setting applies only to aggregate storage databases.

Syntax

```
AUTOMERGEMAXSLICENUMBER n
```

n—Specifies the maximum number of incremental data slices that can exist in a data load without activating the process of automatically merging incremental data slices.

- When the number of incremental data slices is equal to (=) or less than (<) *n*, the incremental data slices are not merged.
- When the number of incremental data slices is greater than (>) *n*, the auto-merge process is activated.

The default value is 4.

During the auto-merge process, Essbase determines the maximum size, as a percentage, that any one incremental data slice can contribute to the maximum number of incremental input cells. Essbase counts the number of cells in all committed incremental data slices. Assume that *r* represents the maximum percentage. If the size of an incremental data slice, as a percentage, is:

- Equal to or less than *r*, the incremental data slice is added to the list of incremental data slices to be automatically merged
- Greater than *r*, the incremental data slice is not added to the list of incremental data slices to be automatically merged

Example

```
AUTOMERGEMAXSLICENUMBER 5
```

Activates the incremental data slice auto-merge process when the number of incremental data slices exceeds 5.

See Also

AUTOMERGE

CALCCACHE

Specifies whether Essbase uses a calculator cache when calculating the database.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCCACHE [appname [dbname]] TRUE | FALSE
```

- *appname*—Optional. Specifies the application for which the setting applies.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.

  If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on the Essbase instance.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which the setting applies.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.

- TRUE—Essbase uses a calculator cache when calculating the database. This is the default.
- FALSE—Essbase does not use a calculator cache when calculating the database.

Description

Essbase uses the calculator cache to create and track data blocks during calculation. Using the calculator cache significantly improves calculation performance. The size of the performance improvement depends on the database configuration.

If required during a calculation, you can override this default setting using the SET CACHE command in a calculation script.

You can specify the size of the calculator cache using the SET CACHE command in a calculation script and the CALCCACHE {HIGH | DEFAULT | LOW} configuration settings.

When the CALCCACHE setting is set to TRUE, Essbase uses the calculator cache, providing that:

- The database has at least two sparse dimensions.
- You calculate at least one full sparse dimension (unless you specify the CALCCACHE ALL option in a calculation script).

Example

```
CALCCACHE Sample Basic FALSE
```

CALCCACHEDEFAULT

Sets a default value for the calculation script SET CACHE command.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCCACHEDEFAULT [appname [dbname]] n
```

- *appname*—Optional. Specifies the application for which the setting applies.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.

  If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on Essbase Server.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which the setting applies.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- *n*—The default calculator cache size, in bytes. If you do not set a default value, it is 200,000 bytes.

Description

Essbase uses the calculator cache to create and track data blocks during calculation. Using the calculator cache significantly improves calculation performance. The size of the performance improvement depends on the database configuration.

You can specify whether Essbase uses a calculator cache by default using the CALCCACHE setting. If required during a calculation, override this default setting using the SET CACHE command in a calculation script.

Example

Assume the Essbase configuration specifies these settings:

```
CALCCACHEHIGH  1000000
CALCCACHEDEFAULT 300000
CALCCACHELOW  200000
```

> **Note:**
>
> In the Essbase configuration, a parameter is not followed by a semicolon; in a calculation script, a parameter must be followed by a semicolon.

You could then use the following SET CACHE commands in a calculation script:

```
SET CACHE HIGH;
```

Sets a calculator cache of 1,000,000 bytes for the duration of the calculation script.

```
SET CACHE DEFAULT;
```

Sets a calculator cache of 300,000 bytes for the duration of the calculation script.

```
SET CACHE LOW;
```

Sets a calculator cache of 200,000 bytes for the duration of the calculation script.

CALCCACHEHIGH

Sets the high value for the calculation script SET CACHE command.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCCACHEHIGH [appname [dbname]] n
```

- *appname*—Optional. Specifies the application for which the setting applies.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.

  If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on Essbase Server.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which the setting applies.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- *n*—The maximum calculator cache size, in bytes (not to exceed 200,000,000 bytes).

Description

Essbase uses the calculator cache to create and track data blocks during calculation. Using the calculator cache significantly improves calculation performance. The size of the performance improvement depends on the database configuration.

You can specify whether Essbase uses a calculator cache by default using the CALCCACHE TRUE | FALSE configuration setting. If required during a calculation, override this default setting using the SET CACHE command in a calculation script.

Example

Assume the Essbase configuration contains these settings:

```
CALCCACHEHIGH  1000000
CALCCACHEDEFAULT 300000
CALCCACHELOW  200000
```

> **Note:**
>
> In the Essbase configuration, a parameter is not followed by a semicolon; in a calculation script, a parameter must be followed by a semicolon.

You could use the following SET CACHE calculator commands in a calculation script:

```
SET CACHE HIGH;
```

Sets a calculator cache of 1,000,000 bytes for the duration of the calculation script.

```
SET CACHE DEFAULT;
```

Sets a calculator cache of 300,000 bytes for the duration of the calculation script.

```
SET CACHE LOW;
```

Sets a calculator cache of 200,000 bytes for the duration of the calculation script.

CALCCACHELOW

Sets the low value for the calculation script SET CACHE command.

This setting does not apply to aggregate storage databases.

Syntax

```
CALCCACHELOW [appname [dbname]] n
```

- *appname*—Optional. Specifies the application for which the setting applies.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.

  If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on Essbase Server.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which the setting applies.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- *n*—The minimum calculator cache size, in bytes.

Description

Essbase uses the calculator cache to create and track data blocks during calculation. Using the calculator cache significantly improves calculation performance. The size of the performance improvement depends on the database configuration.

You can specify whether Essbase uses a calculator cache by default using the CALCCACHE configuration setting. If required during a calculation, override this default setting using the SET CACHE command in a calculation script.

Example

Assume the Essbase configuration specifies these settings:

```
CALCCACHEHIGH  1000000
CALCCACHEDEFAULT 300000
CALCCACHELOW  200000
```

> **Note:**
>
> In the Essbase configuration, a parameter is not followed by a semicolon; in a calculation script, a parameter must be followed by a semicolon.

You could then use the following SET CACHE commands in a calculation script:

```
SET CACHE HIGH;
```

Sets a calculator cache of 1,000,000 bytes for the duration of the calculation script.

```
SET CACHE DEFAULT;
```

Sets a calculator cache of 300,000 bytes for the duration of the calculation script.

```
SET CACHE LOW;
```

Sets a calculator cache of 200,000 bytes for the duration of the calculation script.

CALCLIMITFORMULARECURSION

When set to true, prevents the server from going beyond 128 formula execution levels.

Syntax

```
CALCLIMITFORMULARECURSION TRUE | FALSE
```

- TRUE—Imposes a limit of 128 on the number of formula execution levels. This is the default.
- FALSE—Imposes no limit on the number of formula execution levels.

Description

CALCLIMITFORMULARECURSION limits the number of execution levels of Essbase formulas. If a calculation involves formulas referencing one or more members from sparse dimensions and there are formulas along dense dimension members, the formula execution may be recursive (have multiple execution levels). Formulas with excessive execution levels may crash the server. Setting CALCLIMITFORMULARECURSION to TRUE prevents excessive execution levels from crashing the Essbase Server.

If a formula reaches 128 execution levels and CALCLIMITFORMULARECURSION is set to TRUE (or default), Essbase stops processing that formula and writes error messages in the application log. If a formula reaches 128 execution levels and CALCLIMITFORMULARECURSION is set to FALSE, Essbase continues processing that formula and writes an information message in the application log.

> **Note:**
>
> This setting does not affect formulas in MDX queries (for example, calculated members).

Example

```
Payroll / @SUMRANGE(Payroll, @IRDESCENDANTS(Market))
```

If you added a member named Payroll2 to the Measure dimension in Sample.Basic and used the following formula to calculate it, you would get a recursion error if Market has more than 128 members:

CALCPARALLEL

Enables parallel calculation, defining the number of processing threads.

Syntax

```
CALCPARALLEL [appname [dbname]] n
```

- *appname*—Optional. Specifies that parallel calculation applies to all cubes on the application. If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all cubes in the application. If you do not specify an application, you cannot specify a cube, and the setting applies to all applications and cubes.
- *dbname*—Optional. Specifies that parallel calculation applies only to this cube. If you specify a value for *dbname* but do not include *appname*, the parameter is ignored, and parallel calculation is enabled for all applications and cubes.
- *n*—A required parameter that specifies the number of threads to be made available for parallel calculation.

  - For block storage, an integer between 1-128. The default value, 1, specifies serial calculation: no parallel calculation takes place.
  - For aggregate storage, an integer from 1-128, with 2 the default value.

  A value less than 1 is interpreted as the default size. A value greater than the maximum size is interpreted as the maximum size.

You must restart Essbase to initialize any change to the configuration.

Description

This setting enables CALCPARALLEL parallel calculation. For block storage databases, Essbase analyzes each pass of a calculation to determine whether parallel calculation would optimize the calculation. If it would not, Essbase uses serial calculation even if CALCPARALLEL is set to a number greater than 1.

Notes

- With block storage cubes, Essbase dynamically calculates the number of task dimensions for parallel calculation by starting with a value of 1, determining how many potential tasks are generated, and increasing the number of task dimensions until an optimal limit is reached. If CALCTASKDIMS has been used to increase the number of tasks and to decrease the size of each task identified for parallel calculation, the number of sparse dimensions set with CALCTASKDIMS is used. See Identifying Additional Tasks for Parallel Calculation for more information about what kind of outlines or calculation scripts generate many empty tasks.
- If you increase the number of threads for aggregate storage databases, since the aggregate storage cache is split up amongst the threads, consider increasing the size of aggregate storage memory cache.
- When running a parallel calculation that includes the @XREF calculation function, the application associated with the database returns a timeout error if the number of threads specified for CALCPARALLEL is higher than the number of threads specified by the SERVERTHREADS setting. For example, the default value of SERVERTHREADS is 20. If you set CALCPARALLEL to 25, an application timeout error is generated.
- To learn about a newer type of parallel calculation, see the FIXPARALLEL calculation command.

Example

```
CALCPARALLEL 3
```

Enables up to three threads to perform calculation tasks at the same time.

See Also

Using CALCPARALLEL Parallel Calculation

SET CALCPARALLEL

SET CALCTASKDIMS

SERVERTHREADS

FIXPARALLEL...ENDFIXPARALLEL

CALCTASKDIMS

Specifies the number of sparse dimensions included in the identification of tasks for parallel calculation.

This setting does not apply to aggregate storage cubes.

Syntax

```
CALCTASKDIMS [appname [dbname]] n
```

- *appname*—Optional. CALCTASKDIMS applies to all cubes on the application. If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all cubes in the application. If you do not specify an application, you cannot specify a cube, and the setting applies to all applications and cubes.
- *dbname*—Optional. Cube name to which CALCTASKDIMS applies. If you specify a value for *dbname* but do not include *appname*, the parameter is ignored and the setting applies to all applications and cubes.
- *n*—Required. An integer specifying the number of sparse dimensions to be included when Essbase identifies tasks that can be performed at the same time.

  A value of 1 indicates that only the last sparse dimension in the outline is used to identify tasks. A value of 2, for example, indicates that the last and second-to-last sparse dimensions in the outline are used. Because each unique combination of members from selected sparse dimensions is a potential task, the potential number of parallel tasks is the product of the number of members of the selected dimensions. The maximum value is the number of sparse dimensions in the outline.

  Any value less than 1 is interpreted as 1, any value greater than the number of sparse dimensions in the outline is converted to the largest valid value.

You must restart Essbase to initialize any change to the configuration.

Description

CALCTASKDIMS specifies how many of the sparse dimensions in an outline are used to identify potential tasks that can be run in parallel.

Notes

- If you do not notice an improvement in performance after increasing the value of CALCTASKDIMS, see the note in the SET CALCTASKDIMS topic.
- Use this configuration setting only if your outline generates many empty tasks, thus reducing opportunities for parallel calculation. See Identifying Additional Tasks for Parallel Calculation for more information about what kind of outlines or calculation scripts generate many empty tasks.

Example

```
CALCTASKDIMS Sample Basic 2
```

Specifies that for application Sample and database Basic, the last two sparse dimensions in an outline will be used to identify potential tasks to perform at the same time during a calculation pass.

See Also

CALCPARALLEL

SET CALCPARALLEL calculation command

SET CALCTASKDIMS calculation command

CALCTRACE

This application configuration parameter enables calculation tracing to help debug calculation scripts.

The tracing is done on the cell specified by using the SET TRACE calculation command, or by selecting the cell in Smart View. The output is available in Smart View, as well as in a file, `calc_trace.txt`, located in the cube directory.

Syntax

```
CALCTRACE OFF | ON
```

- OFF – Calculations are not traced. Any SET TRACE commands in calculation scripts are ignored.
- ON – Calculations can be traced. You can specify a cell to be traced in Smart View by selecting a cell in the grid before executing a calculation script. You can also use SET TRACE commands in calculation scripts if you need to trace multiple cells.

Example

```
CALCTRACE ON
```

See Also

SET TRACE

Trace Calculations

CUSTOMCALCANDALLOCTHRUINSERT

Enables execution of aggregate storage custom calculations and allocations through MDX Insert.

Syntax

```
CUSTOMCALCANDALLOCTHRUINSERT [appname [dbname]] TRUE | FALSE
```

- *appname*—Optional. Specifies the aggregate storage application to which the configuration applies.
- *dbname*—Optional. Specifies the aggregate storage cube to which the configuration applies.
- TRUE—The execution of aggregate storage custom calculations and allocations goes through MDX Insert .
- FALSE—Custom calculations and allocations do not execute through MDX Insert. This is the default.

See Also

- Performing Custom Calculations and Allocations on Aggregate Storage Databases
- USE_MDX_INSERT

DATACACHESIZE

Defines the value for the data cache size for Essbase databases. The data cache is a buffer in memory that holds data blocks. Essbase allocates this memory during data load, calculation, and retrieval operations, as needed.

This setting does not apply to aggregate storage databases.

Syntax

```
DATACACHESIZE n
```

n—An integer value expressed in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G):

- Minimum value: 3 megabytes (3 M)
- Maximum value: 2 gigabytes (2 G)
- Default value: 3 megabytes (3 M)

If a value is given without a B, K, M, or G qualifier, it is assumed the value is in bytes.

The qualifier can be in upper or lowercase and can be entered adjacent to the value (10M) or separated by a space (10 M).

Description

DATACACHESIZE specifies, in bytes, kilobytes, megabytes, or gigabytes, the size of the data cache for databases.

Example

```
DATACACHESIZE 90M
```

Sets the data cache size of databases to 90 megabytes.

DEFAULTVIEWBUILD

Sets whether default aggregate view selection and materialization are recreated automatically in the following events:

- when there is a metadata change that would require redesign or rebuilding of aggregate views
- when aggregate views are removed
- when data is loaded to an empty cube

This configuration applies only to aggregate storage cubes. Use the DEFAULTVIEWBUILDSIZE configuration setting to limit the growth of cubes as of a result of automatic aggregations.

If this setting is not enabled, default aggregate views are created by Essbase using internal analysis based on data sampling.

Syntax

```
DEFAULTVIEWBUILD appname TRUE | FALSE
```

- *appname*—Application name. The configuration applies to all cubes within the named application.
- TRUE—Aggregate view selection and materialization are automated.
- FALSE—Aggregate view selection and materialization are not automated.

Example

```
DEFAULTVIEWBUILD TRUE
```

DEFAULTVIEWBUILDSIZE

Sets the maximum allowed cube growth ratio when aggregate view selection and materialization are automated.

This configuration applies only to aggregate storage cubes, when DEFAULTVIEWBUILD configuration is set to TRUE.

Syntax

```
DEFAULTVIEWBUILDSIZE appname n
```

- *appname*—Application name. The configuration applies to all cubes within the named application.
- *n*—Total size ratio limiting the resulting aggregation views when DEFAULTVIEWBUILD configuration is enabled. The maximum growth of the aggregated cube must not exceed this ratio. For example, setting the the total size ratio to `1.3` means that the cube can grow by no more than 30% as a result of the automated aggregation.

Example

```
DEFAULTVIEWBUILDSIZE ASOSamp 1.3
```

DLSINGLETHREADPERSTAGE

Instructs Essbase to load data using a single thread per processing stage, or to use the thread values specified in the DLTHREADSPREPARE and DLTHREADSWRITE configuration settings. By working with these three configuration settings, you may be able to test and improve data load performance.

You can specify this setting for individual databases, for all databases within an application, or for all applications and databases on the server.

Syntax

```
DLSINGLETHREADPERSTAGE [appname [dbname]] TRUE | FALSE
```

- *appname*—Application name. Optional parameter for applying the TRUE or FALSE setting to one or all databases within the application. If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application. If you do not specify an application, you cannot specify a database and the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Database name. Optional parameter for applying the TRUE or FALSE setting to a specific database within the specified application. If you do not specify a value for *dbname*, the setting applies to all databases within the specified application. If *appname* is not specified, you cannot specify *dbname*.
- TRUE—Tells Essbase **not** to use the values in the DLTHREADSPREPARE and DLTHREADSWRITE configuration settings when it performs a data load. Consequently, it performs all data load processes in single-thread stages.
- FALSE—Tells Essbase to use the thread values specified in the configuration settings DLTHREADSPREPARE and DLTHREADSWRITE as the numbers of threads to use in the preparation and write stages of data load processing. The default value is FALSE.

Description

This setting, and related settings DLTHREADSPREPARE and DLTHREADSWRITE, are related to parallel data load processing. Data load processing is divided up into stages that are performed by Essbase using separate processing threads for each stage. By default, a single thread is used for each stage. Taking advantage of the multithreading capabilities of the server machine, the separate single-thread stages can be performed in parallel.

To improve data load performance by maximizing use of processor resource for your situation, you can use these settings to enable additional multiple-thread processing within the preparation and write stages of data load processing.

Notes

- While testing thread values for the DLTHREADSPREPARE and DLTHREADSWRITE configuration settings, you can use the DLSINGLETHREADPERSTAGE setting to quickly revert to using a single thread per stage.
- Enabling use of multiple threads during the preparation and write stages may produce little if any benefit on a single-processor machine.
- Optimizing factors such as the content and organization of the data source can enhance performance more than increasing the numbers of threads to be used. See Optimizing Data Loads.

Examples

Example 1

```
DLSINGLETHREADPERSTAGE Sample Basic TRUE
DLTHREADSPREPARE Sample Basic 3
DLTHREADSWRITE Sample Basic 4
```

Essbase ignores any values specified by DLTHREADSPREPARE and DLTHREADSWRITE while loading data to the Sample Basic application and database. As a result, Essbase uses single threads in each stage.

Example 2

```
DLSINGLETHREADPERSTAGE FALSE
DLTHREADSPREPARE Sample Basic 3
DLTHREADSWRITE Sample Basic 4
```

Based on the first setting, Essbase uses the number of threads specified by the DLTHREADSPREPARE and DLTHREADSWRITE configuration settings for all databases on the server. The settings on the second and third lines specify use of 3 processing threads for the preparation stages and 4 processing threads for the write stages when loading the Sample Basic application and database. Assuming that there are no further related settings, the default value 1 (one) is assumed for all other applications and databases on the server.

Example 3

```
DLSINGLETHREADPERSTAGE Sample FALSE
DLTHREADSWRITE Sample Basic 3
DLTHREADSWRITE Sample Interntl 4
```

In this example Essbase uses the number of threads specified by the DLTHREADSPREPARE and DLTHREADSWRITE

configuration settings for all databases within the application named Sample. To enable usage of different numbers of threads for the write stage for the two different databases, two DLTHREADSWRITE settings are included with different thread values for each specific database. Because no DLTHREADSPREPARE setting is specified, the preparation stage is single-threaded.

DLTHREADSPREPARE

Specifies how many threads Essbase may use during the data load preparation stage, which organizes the source data in memory in preparation for storing the data into blocks. Multiple threads, processing in parallel, may improve data load performance.

You can specify this setting for individual databases, for all databases within an application, or for all applications and databases on the server.

In order for Essbase to use the value specified for this setting, the DLSINGLETHREADPERSTAGE setting must be set to FALSE.

Syntax

```
DLTHREADSPREPARE [appname [dbname]] n
```

- *appname*—Application name. Optional parameter for using the specified number of threads in one or all databases within the application. If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application. If you do not specify an application, you cannot specify a database and the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Database name. Optional parameter for using the specified number of threads when loading the specified database within the specified application. If you do not specify a value for *dbname*, the setting applies to all databases within the specified application. If *appname* is not specified, you cannot specify *dbname*.
- *n*—The number of threads the data load process may use for preparing the data to be loaded. Specify an integer between 1 and 32. The default value is 1.

  If *n* is greater than the maximum or a negative number, the value is assumed to be 32.

Description

This setting, and related settings DLTHREADSWRITE and DLSINGLETHREADPERSTAGE, are related to parallel data load processing. The concept of a *pipeline* is relevant to Essbase data loads. A pipeline is a series of data processing elements in memory that may be executed serially or in parallel. An Essbase data load operation uses a pipeline consisting of 5 stages. By default, a single thread is used for each stage. Therefore, all data load operations need a minimum of 5 threads.

To improve data load performance by maximizing use of processor resource for your situation, you can use these settings to enable additional multiple-thread processing within the preparation and write stages of data load processing. For more information about parallel thread processing in data loads, see Optimizing Data Loads.

Notes

- You can use another configuration setting, DLTHREADSWRITE, to specify the number of threads for the write stage of data load processing.
- Many factors affect the possible optimal values for DLTHREADSPREPARE including the number of processors on the machine and the number of other processes running on the machine. If you want to set this setting to a value higher than the default (1), check with your system administrator, as higher values can consume considerable system resources. As a rule of thumb, do not expect performance advantages if the number of threads for this setting is greater than the number of processors on the server machine.
- Setting the value for DLTHREADSPREPARE to be greater than 1 (one) may produce little if any benefit on a single-processor machine.

Example

```
DLSINGLETHREADPERSTAGE Sample Basic FALSE
DLTHREADSPREPARE Sample Basic 3
```

Because DLSINGLETHREADPERSTAGE is set to FALSE for the Sample Basic application and database, Essbase uses 3 parallel threads during the preparation stage when loading data to Sample Basic.

See Also

DLTHREADSWRITE

DLSINGLETHREADPERSTAGE

DLTHREADSWRITE

Specifies how many threads Essbase may use during the stage of the data load process that writes blocks on the disk. Multiple threads, processing in parallel, may improve data load performance.

Since Essbase uses a single thread during the write stage of the aggregate storage data load process, this setting does not apply to aggregate storage databases.

Syntax

```
DLTHREADSWRITE [appname [dbname]] n
```

- *appname*—Application name. Optional parameter for using the specified number of threads in one or all databases within the application. If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application. If you do not specify an application, you cannot specify a database and the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Database name. Optional parameter for using the specified number of threads when loading the specified database within the specified application. If you

do not specify a value for *dbname*, the setting applies to all databases within the specified application. If *appname* is not specified, you cannot specify *dbname*

- *n*—The number of threads the data load process may use for writing data blocks to the disk. Specify an integer between 1 and 32. The default value is 1. If *n* > 32, or a negative number, the value is assumed to be 32.

Description

This setting, and related settings DLTHREADSPREPARE and DLSINGLETHREADPERSTAGE, are related to parallel data load processing. The concept of a *pipeline* is relevant to Essbase data loads. A pipeline is a series of data processing elements in memory that may be executed serially or in parallel. An Essbase data load operation uses a pipeline consisting of 5 stages. By default, a single thread is used for each stage. Therefore, all data load operations need a minimum of 5 threads.

To improve data load performance by maximizing use of processor resource for your situation, you can use these settings to enable additional multiple-thread processing within the preparation and write stages of data load processing.

You can specify DLTHREADSWRITE for individual databases, all databases within an application, or for all applications and databases on the server.

In order for Essbase to use the value specified for DLTHREADSWRITE, the configuration setting DLSINGLETHREADPERSTAGE must be set to FALSE.

For more information about parallel thread processing in data loads, see Optimizing Data Loads.

Notes

- You can use another configuration setting, DLTHREADSPREPARE, to specify the number of threads for the preparation stage of data load processing.
- Many factors affect the possible optimal values for DLTHREADSWRITE including the number of processors on the machine and the number of other processes running on the machine. If you want to set this setting to a value higher than the default (1), check with your system administrator, as higher values can consume considerable system resources. As a rule of thumb, do not expect performance advantages if the number of threads for this setting is greater than the number of processors on the server machine.
- Setting the value for DLTHREADSWRITE to be greater than 1 (one) may produce little if any benefit on a single-processor machine.

Example

```
DLSINGLETHREADPERSTAGE Sample Basic FALSE
DLTHREADSWRITE Sample Basic 3
```

Because DLSINGLETHREADPERSTAGE is set to FALSE for the Sample Basic application and database, Essbase uses 3 parallel threads during the write stage when loading data to Sample Basic.

See Also

DLTHREADSPREPARE

DLSINGLETHREADPERSTAGE

DYNCALCCACHEMAXSIZE

Specifies the maximum amount of memory allocated for the dynamic calculator cache for each database. The specified value takes effect for all databases that are opened after the server is started.

The dynamic calculator cache is a memory buffer that holds data blocks that are expanded to include dynamically calculated members. Essbase allocates memory in the dynamic calculator cache to store these blocks during retrievals or calculations that involve dynamically calculated members.

Using dynamic calculator cache may improve retrieval performance by reducing the number of calls to the operating system to do memory allocations.

This setting does not apply to aggregate storage databases.

Syntax

```
DYNCALCCACHEMAXSIZE [appname [dbname]] n
```

- *appname*—If you specify an application name, the setting applies to all databases within the application. If you do not specify an application name, the setting applies to all applications and databases on the server.
- *dbname*—If you specify a database name, the setting applies only to the database. If you do not also specify an application name, the setting applies to all applications and databases on the server.
- *n*—An integer expressed in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G)
  - Minimum value: 0 megabytes (0 M). If the value is 0, Essbase does not use dynamic calculator cache.
  - Default value: 20 megabytes (20M, which is 20,971,520 bytes)
  - The maximum amount of memory that can be allocated is 256 GB:
  - If a value is given without a B, K, M, or G qualifier, it is assumed the value is in bytes.
  - The qualifier can be in upper or lowercase and can be entered adjacent to the value (10M) or separated by a space (10 M).

Example

```
DYNCALCCACHEMAXSIZE 30M
```

Sets 30 megabytes as the maximum size for the dynamic calculator cache.

ENABLERTSVLOGGING

Determines whether Essbase logs runtime substitution variables that are used in a calculation script.

Runtime substitution variable log entries are written to the application log file.

Syntax

```
ENABLERTSVLOGGING [appname [dbname]] TRUE | FALSE
```

- *appname*—Optional. Specifies the application for which runtime substitution variable logging is to be set.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.

  If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on Essbase Server.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which runtime substitution variable logging is to be set.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- TRUE—Runtime substitution variables that are used in a calculation script are logged. For information about the format of these log entries, see Logging Runtime Substitution Variables.
- FALSE—Runtime substitution variables that are used in a calculation script are not logged. The default value is FALSE.

Example

```
ENABLERTSVLOGGING TRUE
```

See Also

SET RUNTIMESUBVARS calculation command

FORCEALLDENSECALCON2PASSACCOUNTS

Normally, a two-pass tagged member of a dense accounts dimension triggers a second calculation pass on all dense cells of the data block. The false parameter value for this setting blocks the second pass for all other than the cells for the member tagged as two-pass.

Syntax

```
FORCEALLDENSECALCON2PASSACCOUNTS TRUE | FALSE
```

- TRUE—(Default value) When a two-pass member of a dense accounts dimension is calculated, the second calculation pass calculates all dense cells of the data block.
- FALSE—In the same situation, the FALSE setting blocks the second calculation pass for all dense cells except those affiliated with the two-pass member.

Description

This setting addresses the situation where a two-pass member of a dense accounts dimension links through @XREF to a two-pass member of a dense accounts dimension in another database outline, and that two-pass member links back to the original outline. The additional calculations in the second calculation pass can result in an infinite loop. The FALSE parameter value blocks the additional calculations. If you are very cautious about data correctness, check calculation results.

Example

```
FORCEALLDENSECALCON2PASSACCOUNTS FALSE
```

FORCESHUTDOWNINTERVAL

This setting applies to block storage and aggregate storage databases.

The default interval is 10 seconds.

Syntax

```
FORCESHUTDOWNINTERVAL n
```

Example

```
FORCESHUTDOWNINTERVAL 2000
```

Checks for a heartbeat every 2000 seconds.

GRIDEXPANSION

When set to ON, improves performance when transparent partitions are queried.

Syntax

```
GRIDEXPANSION [appname [dbname]] ON | OFF
```

- *appname*—Optional. If you specify an application name, the setting applies to all databases within the named application. If you do not specify an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you specify a database name and an application name, the setting applies only to the named database. If you do not also specify an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.

- ON—This is the default value. Enables grid expansion.
- OFF— Suppresses grid expansion.

Description

GRIDEXPANSION improves performance of some queries. If all of the following conditions are met, however, client queries may receive incorrect results (such as most data values displaying as #MISSING, whether or not cells contain data):

- The client queries the target database of a transparent partition.
- The client query requests values from a dynamically calculated block.
- Cells requested from the dynamically calculated block reference dense, dynamically calculated members.
- Dense, dynamically calculated members depend on values from one or more source databases.

See Also

GRIDEXPANSIONMESSAGES

GRIDEXPANSIONMESSAGES

Sets whether grid expansion-related messages are displayed to Smart View and other grid client users, and are written to the application log.

Syntax

```
GRIDEXPANSIONMESSAGES [appname [dbname]] ON | OFF
```

- *appname*—Optional. If you specify an application name, the setting applies to all databases within the named application. If you do not specify an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you specify a database name and an application name, the setting applies only to the named database. If you do not also specify an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- ON—Allows grid-expansion-related messages.
- OFF—This is the default value. Suppresses grid-expansion-related messages.

Description

If a grid client user retrieves data from a partition, the following message may be displayed repeatedly and written to the application log:

```
Grid expansion enabled for this query
```

To prevent this message from appearing, set GRIDEXPANSIONMESSAGES to OFF.

Example

```
GRIDEXPANSIONMESSAGES OFF
```

See Also

GRIDEXPANSION

GRIDSUPPRESSINVALID

Sets whether invalid attribute combinations, which are represented on the grid by #invalid, are suppressed in Smart View. An invalid attribute combination is the result of an intersection of a dimension member for which an attribute is not assigned or, if an attribute is assigned to the member, the attribute combination is not within the scope of the grid query or the assigned attribute is incorrect. Invalid attribute combinations are suppressed when the row contains all #invalid values. Valid combinations with #MISSING values are not suppressed.

This configuration setting applies to block storage and aggregate storage databases.

Syntax

```
GRIDSUPPRESSINVALID [appname [dbname]] TRUE | FALSE
```

- *appname*—Optional. If you specify an application name, the setting applies to all databases within the named application. If you do not specify an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you specify a database name and an application name, the setting applies only to the named database. If you do not also specify an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- TRUE—Enables suppressing invalid attribute combinations on the grid. This is the default value.
- FALSE—Invalid attribute combinations are not suppress on the grid.

Example

```
GRIDSUPPRESSINVALID Sample Basic TRUE
```

Suppresses #invalid values in the Sample.Basic database.

See Also

Suppressing Invalid Attribute Combinations in the Grid.

HYBRIDBSOINCALCSCRIPT

Controls whether cubes in the application use hybrid mode in calculation scripts when stored members depend on dynamic members. When set to FULL, the calculation engine uses hybrid mode to calculate the results, and then stores them.

Hybrid mode means that wherever possible, data calculation executes with efficiency similar to that of aggregate storage databases.

This setting is not applicable to aggregate storage databases.

If you enable this setting, do not disable ASODYNAMICAGGINBSO, which is on by default (meaning hybrid mode is enabled for queries).

Syntax

```
HYBRIDBSOINCALCSCRIPT [appname [dbname]] FULL|NONE
```

- *appname*—Optional. Specifies the application for which hybrid mode is used.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which hybrid mode is used.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- FULL—Calculation scripts run in hybrid mode.
- NONE—Calculation scripts run in block storage mode. This is the default.

Notes

The following limitations apply to hybrid mode for calculation scripts. If encountered, Essbase defaults to block storage execution for these kinds of calculation scripts.

- CALC DIM, CALC ALL, AGG, and any other assignment-free expressions that calculate a sub-tree, do not use hybrid mode.

  Oracle recommends limiting your use of CALC DIM and AGG to dimensions wherein no stored members are dependent on dynamic members. To calculate upper-level stored members that depend on dynamic members, use assignment formulas with calculation functions.
- DATAEXPORT for dynamic members does not use hybrid mode.
- Intelligent calculation does not use hybrid mode.
- Do not use CREATENONMISSINGBLOCK or CREATEBLOCKONEQ in calculation scripts you want to run in hybrid mode.
- CALCPARALLEL is not supported in hybrid mode. For parallel calculation, use FIXPARALLEL.

Example

```
HYBRIDBSOINCALCSCRIPT FULL
```

See Also

ASODYNAMICAGGINBSO

Adopt Hybrid Mode for Fast Analytic Processing

SET HYBRIDBSOINCALCSCRIPT

IDLETIMEMERGE

Enables the performing of slice merge operations while the server is idle. This allows incremental slices that were created with new data during aggregate storage data load, to be merged soon after load, during server idle time.

Syntax

```
IDLETIMEMERGE  TRUE | FALSE
```

The default value is false.

Description

IDLETIMEMERGE specifies whether slice merge is enabled and performed during server idle time.

Use this setting to enable or disable continuous slice merge.

Example

```
IDLETIMEMERGE  TRUE
```

Sets a continuous slice merge to be enabled during server idle time.

See Also

SVRIDLETIME

IGNORECONSTANTS

Controls whether #Missing values, when used as operands in formulas, should remain #Missing after the formula calculation.

Syntax

```
IGNORECONSTANTS TRUE [BOTTOMUP] | FALSE
```

- TRUE—Default option. #Missing values remain missing regardless of interaction with formula constants.
- BOTTOMUP—If used after TRUE, this parameter additionally causes query execution to occur in bottom-up mode, to resolve dependency analysis quickly in cases where the formula cache has a relatively small input data set.
- FALSE— #Missing values can be changed by interaction with formula constants.

Description

If a #Missing data value is processed in a formula with a constant or other data-independent construct, the default behavior is that #Missing is not treated like a data value. For example, if A is missing, A+5 returns #Missing.

If you set IGNORECONSTANTS to FALSE, #Missing is treated like a data value. For example, if A is missing, A+5 returns 5.

To optimize cubes in the application for hybrid mode for faster formula execution, you can use this setting configured as follows:

```
IGNORECONSTANTS TRUE BOTTOMUP
```

The above usage is recommended for hybrid mode performance tuning if both of these are true:

- you experience query performance problems
- formulas are complex and contain many cross-dimensional operators or IF/ELSE statements

Complex formulas can cause the formula cache to grow large while also being sparse (having a relatively small input data set). Using IGNORECONSTANTS TRUE BOTTOMUP forces query execution to occur in bottom-up mode, to resolve dependency analysis more efficiently in cases where the formula cache is sparse.

Example

If the configuration is as follows:

```
IGNORECONSTANTS TRUE
```

then the result for X in the following formula is #Missing

```
IF(X)
5;
ELSE
3
ENDIF
```

See Also

@NONEMPTYTUPLE calculation function

NONEMPTYTUPLE property in MDX Optimization Properties

INDEXCACHESIZE

Defines the value for the index cache size for Essbase databases. The index cache is a buffer in memory that holds index pages. Essbase allocates this memory at startup of the database.

The value of the index cache size can be expressed in bytes, kilobytes, megabytes, or gigabytes. Terabytes must be expressed in gigabytes.

This setting does not apply to aggregate storage databases.

Syntax

```
INDEXCACHESIZE n
```

n—An integer value expressed in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G):

- Minimum value: 1 megabytes (1 M)
- Maximum value: 256 TB

  Default value for buffered I/O: 1 megabyte (1 M)

If a value is given without a B, K, M, or G qualifier, it is assumed the value is in bytes.

The qualifier can be in upper or lowercase and can be entered adjacent to the value (10M) or separated by a space (10 M).

Example

```
INDEXCACHESIZE 100M
```

Sets the index cache size of databases to 100 megabytes.

LONGCALCTIMETHRESHOLD

Log informational messages for long-running commands in Essbase calculation scripts.

Syntax

```
LONGCALCTIMETHRESHOLD n
```

Where *n* is the upper-limit execution time, in seconds, before which Essbase tracks the calculations and logs alert messages. Default: 0 seconds (calculation tracking is off).

Description

Use this parameter to specify the upper-limit execution time, in seconds, that top-level calculation command blocks can run before being considered long running and thus subject to tracking. A top-level command block is a command block that is not enclosed in a FIX/ENDFIX block. Essbase sends alert messages if any top level command block runs longer than the specified time limit.

The logged messages include the following information.

- CALC_USER: ID of the user running the calculation script
- CALC_SCRIPT_NAME: Calculation script name
- LINE_NUMBERS: Range of script line numbers encompassed by the calculation command
- Calc Command Text: First 500 characters of the calculation command code
- BLOCKS_CREATED: Number of blocks created by the calculation command
- BLOCKS_READ: Number of blocks read into memory during execution of the calculation command
- BLOCKS_WRITE: Number of blocks updated by the calculation command

- EXEC_TIME: Time, in milliseconds, that the calculation command has been running
- Statistics about custom-defined functions, if any were executed.

Examples

```
LONGCALCTIMETHRESHOLD 60
```

Specifies message logging for calculations running 60 seconds or longer.

```
LONGCALCTIMETHRESHOLD .001
```

Turns on diagnostics for most calculation commands (the threshold is very small, but not zero).

The following request-pending message indicates that user *essuser* executed calculation script `esscalcx1.csc` on cube *Sample.Basic*. The execution time of the command block from line 9 - 15 was about 75 seconds. There were 562500, 562500, and 1125000 blocks read, written, and created, respectively. The text of the command block is printed inside {} braces.

```
[2019-11-27T17:43:51.783-07:00] [Sample] [NOTIFICATION:16] [CAL-912]
[CAL] [ecid: 1445992953893,0]
[tid: 1107577152] [Basic/essuser] [CALC_USER: essuser]
[CALC_SCRIPT_NAME: esscalcx1.csc] [LINE_NUMBERS: 9 - 15]
[EXEC_TIME: 75420] [BLOCKS_READ: 562500] [BLOCKS_WRITE: 562500]
[BLOCKS_CREATED: 1125000]
Calc Command Text [ FIX ("Conversion Rate","No Period","Plan_A","No
CO","No Agreement","No Supplier",@RELATIVE
("From Group1",0),@RELATIVE("ToCO Group1",0),"No From Currency","No To
Currency") "All Product"
(@CREATEBLOCK("No Product");)ENDFIX]
```

LONGQUERYTIMETHRESHOLD

Capture statistics on long-running grid and MDX queries. This setting lets you specify the lowest query-time length, in seconds, for which you want to capture statistical information.

Syntax

```
LONGQUERYTIMETHRESHOLD n
```

Where *n* is a number of seconds. If set to 0, then this setting is off.

When to Use Different Query Tracing Options

The QUERYTRACE configuration setting is designed for debugging a single query, in case any problems are observed. It prints the trace log to the cube directory as `query_trace.txt`, and the file is cleared by default before each query execution. QUERYTRACE should not be left enabled in the application configuration for long term use, as it may affect performance.

The TRACE_REPORT configuration setting provides fewer details than QUERYTRACE, but is able to trace multiple, concurrent queries. TRACE_REPORT can be enabled for long term use, as it does not affect performance. It is a good option for gathering information on many concurrent queries running over a period of time. TRACE_REPORT prints the trace log to the cube directory as `trace_report.log`, and the file is not cleared between queries. If LONGQUERYTIMETHRESHOLD is enabled, this setting is disabled, and nothing is printed to `trace_report.log`.

The LONGQUERYTIMETHRESHOLD configuration setting is a good choice for production environments. It prints statistics to the application log file about any query that runs longer than a defined number of milliseconds.

Enabling query logging by setting the QUERYLOG parameter in *dbname*`.cfg` (in the cube directory) is designed for tracking user query patterns for a cube, in XML format.

Example

```
LONGQUERYTIMETHRESHOLD 10
```

Specifies statistics collection for queries running 10 seconds or longer.

LONGREQTIMETHRESHOLD

Log informational messages for long-running requests sent to the Essbase application.

Syntax

```
LONGREQTIMETHRESHOLD n
```

Where *n* is the upper-limit execution time, in seconds, before which Essbase tracks the requests and logs alert messages. Default: 600 seconds. Minimum: 60 seconds (or 0 to turn off request tracking).

Description

The logged messages include the following information.

- REQ_ID: a unique number used to identify a request
- REQ_INFO: a string describing the type of request (for example, Calculation, Data-load)
- REQ_STATE: a string describing the current state of the request when this message is logged (for example, **started**, **in_progress**, or **finished**)
- EXEC_TIME: time, in milliseconds, that the request has been running
- LEVEL0_BLOCKS: number of level 0 blocks at the time the message is logged. Essbase only logs this information for requests that could potentially change the cube.
- TOTAL_BLOCKS: number of total blocks at the time the message is logged. Essbase only logs this information for requests that could potentially change the cube.

Example

```
LONGREQTIMETHRESHOLD 600
```

Specifies message logging for requests running 600 seconds or longer.

The following request-pending message in the log shows that request [REQ_ID: 109] has been running for 650232 milliseconds. Therefore, it is considered a long-running request.

```
[2020-01-18T17:31:03.667-08:00] [Sample] [NOTIFICATION:16] [REQ-451]
[REQ] [ecid: 1484789397502,0] [tid: 9664]
[DBNAME: Basic] [REQ_ID: 109] [REQ_INFO: Calculate] [REQ_STATE:
in_progress] [EXEC_TIME: 650232]
```

MAXFORMULACACHESIZE

Applies to aggregate storage databases, or to block storage databases in hybrid mode. Specifies the maximum size of the formula cache to be made available for calculating members with formulas.

Syntax

```
MAXFORMULACACHESIZE [appname [dbname]] n
```

- *appname*—Optional. To set the cache size maximum for a specific application, specifiy the application name.
- *dbname*—Optional. To set the cache size maximum for a specific database, specifiy the database name. If *dbname* is specified, *appname* must also be specified.
- *n*—An integer that specifies the number of kilobytes (KB) to set as the maximum cache size to be made available for calculating members with formulas. The default is 1024.

Description

Essbase administrators can use this setting to limit how much memory may be used by a single query.

If the amount of cache Essbase sets aside for calculating outline members is insufficient, Essbase switches its formula cache mechanism to allocate only existing and temporary cache values. If this occurs and query tracing is enabled, the following message is logged in `query_trace.txt`:

```
Max formula cache size overflow. Calculation cache will use map
structure for values. If query performance is insufficient try to
extend MAXFORMULACACHESIZE over this limit: value
```

The following guidelines can help you determine what value to use for *n*:

1. Identify which queried dimensions are represented by dynamic members.

2. Multiply the sizes of those dimensions to get a number of members.
3. Multiply the number of members by 8 to get the recommended *n* value (not more than 4G).

For example, the default formula cache size of 1024 allows 1024/8=128 members to be in the cache.

Notes

- This setting is only relevant if your query references at least one dynamic member with a formula, or if your MDX query has a calculated member in the WITH section.
- This cache is allocated per calculation thread. Concurrent MDX requests can be allocated multiple cache objects, each with a maximum size specified in MAXFORMULACACHESIZE.
- The entire specified amount is not used unless needed.
- The memory is released after the query completes.

Example

```
MAXFORMULACACHESIZE 2048
```

Sets the aggregate storage formula cache size maximum to 2048 KB for every application and database.

See Also

QUERYTRACE

TRACE_REPORT

MAXLOGINS

Sets a limit on the number of user sessions that can be connected to the Essbase Server at any one time.

Syntax

```
MAXLOGINS n
```

n—Any integer from 1000 to 1048575 is valid. The default value is 10000.

Description

This setting limits the maximum number of user sessions allowed to connect to the Essbase Server at any one time. This number includes multiple instances of the same user.

You may wish to adjust the value of MAXLOGINS to match computer resources, or to more closely manage concurrent ports and user sessions. A concurrent port is used for each unique combination of client machine, Essbase Server and login name. For example, the same user with five open Excel worksheets connected to the same Essbase Server use one port, but five sessions.

Notes

- Increasing the value of MAXLOGINS increases memory use approximately 6 bytes per user session.
- If the setting is less than the minimum value, 1000, the value is assumed to be 1000.

Example

```
MAXLOGINS 50000
```

Increases the maximum number of simultaneous logins possible, from the default of 10000 to 50000.

MAXNUMBEROFACTIVEDB

Specifies the maximum number of active databases that can be accessed concurrently. If the maximum number of active databases is exceeded, the database does not start.

Syntax

```
MAXNUMBEROFACTIVEDB n
```

n—Specifies the maximum number of databases that can be accessed concurrently.

A value of 0 means that there is no maximum limit. The default value is 0.

Example

```
MAXNUMBEROFACTIVEDB 10
```

Specifies that 10 databases can be active.

MAX_REQUEST_GRID_SIZE

Specifies the maximum size of the request grid. The request grid is the number of cells requested from the target (an aggregate storage database) and sent to the data source. Limiting the size of the request grid, which can be millions of cells, ensures a reasonable response time.

If you find that you must set a small request grid size, you should look into improving the design of the application.

Syntax

```
MAX_REQUEST_GRID_SIZE [appname [dbname]] n
```

- *appname*—Optional. Specifies the application for which the request grid size is to be set.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.

  If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on Essbase Server.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which the request grid size is to be set.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored, and logging diagnostic messages is enabled for all applications and databases on Essbase Server.
- *n*—Specifies the size of the request grid to be returned from the data source.

  The default value is 10 million (10000000) cells.

  The maximum value is limited by the unsigned int value of 4294967295.

You must restart Essbase Server to initialize any change to the configuration file.

Example

```
MAX_REQUEST_GRID_SIZE ASOSamp 5000000
```

Limits the request grid to 5 million cells for all databases associated with the ASOSamp application.

See Also

MAX_RESPONSE_GRID_SIZE configuration setting

MAX_RESPONSE_GRID_SIZE

Specifies the maximum size of the response grid. The response grid is the number of cells that the target (an aggregate storage database) sends to the source.

The amount of memory required to temporarily hold the response grid in the data target is proportional to the size of the request grid (MAX_REQUEST_GRID_SIZE). In the case of a huge request grid with millions of cells, the amount of memory required for the response grid to be sent in one operation could pose problems (for example, the system could reach memory boundaries or fail to allocate enough memory). With the MAX_RESPONSE_GRID_SIZE configuration setting, Essbase splits the request grid into slices of data and sends multiple, smaller response grids to the source.

Syntax

```
MAX_RESPONSE_GRID_SIZE [appname [dbname]] n
```

- *appname*—Optional. Specifies the application for which the response grid size is to be set.

If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

To enable the setting for a specific database, you must specify an application and database.

If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on Essbase Server.

- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which the response grid size is to be set.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored, and logging diagnostic messages is enabled for all applications and databases on Essbase Server.

- *n*—Specifies the size of the slice of the response grid to be sent to the data target.

  The default value is one million (1000000) cells, which requires 8 MB of memory.

  For example, if MAX_REQUEST_GRID_SIZE is set to one billion (1000000000) cells and MAX_RESPONSE_GRID_SIZE is set to one million (1000000) cells, the size of the response grid is one thousand (1000) cells.

You must restart Essbase Server to initialize any change to the configuration file.

Example

```
MAX_RESPONSE_GRID_SIZE ASOSamp 500000
```

Limits the response grid to a half-million cells (which requires 4 MB of memory) for all databases associated with the ASOSamp application.

See Also

MAX_REQUEST_GRID_SIZE configuration setting

MDXINSERTBUFFERAGGMETHOD

Defines how the output buffer should be created for an MDX Insert request on the database.

Syntax

```
MDXINSERTBUFFERAGGMETHOD ADD | LAST
```

- LAST—This is the default behavior, if MDXINSERTBUFFERAGGMETHOD is unset. If, during an Insert operation, a value needs to be written to an output buffer location that already contains a value, the latest value overwrites the older value.
- ADD—If, during an Insert operation, a value needs to be written to an output buffer location that already contains a value, the latest value is summed with the older value.

Description

During execution of the MDX Insert query, an output buffer is created in memory which accumulates with values until the query is completed. This setting enables you to define the method with which values are aggregated in the output buffer. The method that you use can have an effect on the data results of the MDX Insert operation.

Assume that in an MDX Insert query, two source tuples are mapped to a single target tuple, as shown:

```
INSERT
([Payroll], [Jan]) TO ([Revised Payroll], [Jan])
([Payroll], [Feb]) TO ([Revised Payroll], [Jan])
...
```

Assume that the value of ([Payroll], [Jan]) is 100, and the value of ([Payroll], [Feb]) is 200.

Using the default buffer aggregation behavior (LAST),

```
MDXINSERTBUFFERAGGMETHOD LAST
```

1. The value for ([Payroll], [Jan]) is written to the output buffer for ([Revised Payroll], [Jan]), making its value 100.
2. The value for ([Payroll], [Feb]) is written to the same output buffer for ([Revised Payroll], [Jan]), overwriting the previous value, and changing it to 200.

If you change the buffer aggregation behavior to ADD,

```
MDXINSERTBUFFERAGGMETHOD ADD
```

1. The value for ([Payroll], [Jan]) is written to the output buffer for ([Revised Payroll], [Jan]), making its value 100.
2. The value for ([Payroll], [Feb]) is added to the same output buffer for ([Revised Payroll], [Jan]), increasing its value to 300.

MDXINSERTREQUESTTIMEOUT

Sets the number of seconds after which Essbase times out an MDX Insert request on the database.

Syntax

```
MDXINSERTREQUESTTIMEOUT n
```

Where *n* is the number of seconds the MDX Insert request is permitted to run before timing out. The default is -1, meaning there is no timeout.

```
MDXINSERTREQUESTTIMEOUT 240
```

Sets the timeout for MDX Insert requests at four minutes.

MDXQRYGOVCOUNT

Initializes a counter (number of check conditions) to control how often Essbase checks for conditions that would warrant termination of an MDX query. Using this counter can reduce or increase the default number of checks (1000); reducing the number of checks (by setting n higher) improves performance. The counter starts at n and decrements until the counter reaches zero: at that time Essbase performs a check.

Syntax

```
MDXQRYGOVCOUNT [appname [dbname]] n
```

- *appname*—Optional. Specifies the application for which to apply the checking counter. If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application. To enable the setting for a specific database, you must specify an application and database.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which to apply the checking counter. If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- *n*—Integer specifying the counter (number of check conditions) that Essbase checks for conditions that warrant query termination. You must specify this parameter or Essbase ignores this setting. If do not specify *appname* or *dbname*, the counter applies to the entire server. The default value is 1000. The minimum value is 100, and the maximum value is 5000.

> **Note:**
>
> You can use the **Esc** key to cancel any query running from MaxL Shell.

Example

```
MDXQRYGOVCOUNT 1500
```

See Also

QRYGOVEXECTIME

QRYGOVEXECBLK

MULTIPLEBITMAPMEMCHECK

Enforces the size limit for the amount of memory that is used for the calculator cache when Essbase selects the multiple bitmap cache option.

This setting does not apply to aggregate storage databases.

Syntax

```
MULTIPLEBITMAPMEMCHECK TRUE | FALSE
```

- TRUE—The size limit is enforced.
- FALSE—The size limit is not enforced.

Description

If the setting is present and its value is TRUE, then any time the memory limit is exceeded for the calculator cache in multiple bitmap cache mode, it will switch to single bitmap mode and enforce the size limit that you selected.

If the setting is not present or has any other value than TRUE, then the limit is not strictly enforced, and your server process may grow too large.

Example

```
MULTIPLEBITMAPMEMCHECK TRUE
```

See Also

CALCCACHE

PARCALCMULTIPLEBITMAPMEMOPT

NUMBLOCKSTOEXTEND

Determines the number of bytes by which data files in block storage databases are extended to accommodate block updates that require additional disk space.

Syntax

```
NUMBLOCKSTOEXTEND [appname [dbname]] n
```

The product of *n* and the currently requested block size is the number of bytes by which the data file is extended.

The default value is 2,048.

Description

When the Essbase block storage kernel updates a block, it writes to a new disk location. The block storage kernel searches free space to find a new disk location to use. If there is not enough free space to service the current request, the data file is extended.

> **Note:**
>
> Upon first upgrading to this release, there is an increase in the amount of disk space pre-allocated for page files unless you set NUMBLOCKSTOEXTEND to 1.

Example

```
NUMBLOCKSTOEXTEND Sample Basic 2240
```

PARCALCMULTIPLEBITMAPMEMOPT

Optimizes memory use when using multiple bitmap mode for the calculator cache during CALCPARALLEL parallel calculation.

This setting does not apply to aggregate storage databases.

Syntax

```
PARCALCMULTIPLEBITMAPMEMOPT TRUE | FALSE
```

- TRUE—Memory usage is optimized when using multiple bitmap mode during CALCPARALLEL parallel calculation.
- FALSE—Memory usage is not optimized. This is the default.

Description

If the setting is present and its value is TRUE, then Essbase optimizes memory usage when using parallel calculation in calculator cache multiple bitmap mode. This setting can be used together with, or separately from, MULTIPLEBITMAPMEMCHECK.

Example

```
PARCALCMULTIPLEBITMAPMEMOPT TRUE
```

See Also

CALCCACHE

CALCPARALLEL

MULTIPLEBITMAPMEMCHECK

QUERYRESULTLIMIT

Sets the maximum number of cells returned by an MDX or grid client query. Applies to block storage, aggregate storage and hybrid aggregation databases.

Syntax

```
QUERYRESULTLIMIT [appname [dbname]] n
```

appname—Optional. Applies the query result limit to the application specified. If you specify *appname*, you must also specify a value for *n*, or Essbase ignores QUERYRESULTLIMIT. If you do not specify an application, you cannot specify a database, and the query result limit applies to all applications and databases on the server. If you specify a value for *appname* and do not specify a value for *dbname*, the query time limit applies to all databases in the specified application.

dbname—Optional. Must be used with *appname* and *n*, or the server ignores QUERYRESULTLIMIT. If you specify *dbname*, *appname*, and *n*, the query result limit is applied only to the specified database.

n—An integer value between 0 and 2^31 specifies the number of query result cells that the server allows a query to return.

The default value is 1000000 (1M).

Description

QUERYRESULTLIMIT specifies the maximum number of result cells that an MDX query or grid client query can retrieve before Essbase terminates the query and returns an error message.

Use this setting to limit the result volume of queries, and prevent a query from freezing when a very large number of result cells are returned.

Examples

```
QUERYRESULTLIMIT Sample Basic 100000
```

Sets 100,000 cells as the maximum number of results cells returned in a query to the Basic database for the Sample application.

```
QUERYRESULTLIMIT 150000
```

Sets 150,000 cells as the maximum number of cells that a query can return before being terminated. The query result limit applies to all applications and databases on the Essbase Server that corresponds to this configuration.

QRYGOVEXECBLK

Sets the maximum number of blocks that a query can access before the query is terminated.

This setting does not apply to aggregate storage databases.

Syntax

```
QRYGOVEXECBLK [appname [dbname]] n
```

- *appname*—Optional. Applies the query block limit to the application specified. If you specify *appname*, you must also specify a value for *n*, or Essbase Server ignores QRYGOVEXECBLK. If you do not specify an application, you cannot specify a database, and the query block limit applies to all applications and databases on the server. If you specify a value for *appname* and do not specify a value for *dbname*, the query time limit applies to all databases in the specified application.
- *dbname*—Optional. Must be used with *appname* and *n*, or Essbase Server ignores QRYGOVEXECBLK. If you specify *dbname*, *appname*, and *n*, the query block limit is applied only to the specified database.
- *n*—The value of *n* specifies the number of blocks that Essbase Server allows a query to access before the query is terminated. You must specify this parameter or the server ignores QRYGOVEXECBLK. If you do not specify *appname* or *dbname*, the query block limit applies to the entire server.

Description

QRYGOVEXECBLK specifies the maximum number of blocks that a query can retrieve before Essbase Server terminates that query (a request for information sent to a database). You can apply this setting to an entire server, to all the databases in a single application, or to a single database.

When a query exceeds the block limit and is terminated, an error message is written to the application log of the application accessed for the query.

Restarting Essbase Server after adding or changing this setting activates the new setting values.

Use QRYGOVEXECBLK to prevent these types of queries:

- A long-running query against a database that accesses atrributes at a high level, forcing many dynamic calculations to occur.
- A query that uses the zoom-in "Drill to bottom" option in a large dimension.
- A query that uses the zoom-in "Drill to all levels" option in a a large dimension.

Use QRYGOVEXECBLK, for example, if you have users who try to retrieve so much data in a single query that their query appears to hang for minutes at a time. A query launched against the database involving attribute dimensions, for example, may be larger than the user realizes.

Notes

- If you use an invalid value (such as a negative number, a letter, a word, or a special character) for *n*, Essbase Server ignores QRYGOVEXECBLK.
- Query governor settings are ignored during data load and calculation. You can leave query governor settings in the configuration file whether you are performing these operations or querying against the data.

Example

```
QRYGOVEXECBLK Sample Basic 3
```

Sets three blocks as the maximum number of blocks that a query to Sample Basic can access before being terminated. A block is created for each unique combination of sparse dimension members. If a user issues a query that accesses four unique combinations of sparse dimensions, Essbase Server terminates the query and writes a message to the application log.

```
QRYGOVEXECBLK 5
```

Sets five blocks as the maximum number of blocks that a query can access before being terminated. The query time limit applies to all applications and databases on Essbase Server .

See Also

QRYGOVEXECTIME

QRYGOVEXECTIME

Sets the maximum amount of time a query can use to retrieve and deliver information before the query is terminated.

Syntax

```
QRYGOVEXECTIME [appname [dbname]] n
```

- *appname*—Optional. Applies the query time limit to the application specified. If you specify *appname*, you must also specify a value for *n*, or Essbase Server ignores QRYGOVEXECTIME. If you do not specify an application, then you cannot specify a database, and the query time limit applies to all applications and databases on Essbase Server. If you specify a value for *appname* and do not specify a value for *dbname*, the query time limit applies to all databases in the specified application.
- *dbname*—Optional. Must be used with *appname* and *n*, or Essbase Server ignores QRYGOVEXECTIME. If you specify *dbname*, *appname*, and *n*, the query time limit is applied only to the specified database.
- *n*—Integer specifying the number of seconds that Essbase Server allows a query to run before the query is terminated. The default value is 300 seconds. If you do not specify *appname* or *dbname*, the query time limit applies to the entire server.

Description

QRYGOVEXECTIME specifies the maximum amount of time that a query can run before Essbase Server terminates the query (a request for information sent to a

database). You can apply this setting to an entire server, to all the databases in a single application, or to a single database.

When a query exceeds the time limit and is terminated, an error message is written to the application log of the application accessed for the query.

Restarting Essbase Server after adding or changing this setting activates the new setting values.

Use QRYGOVEXECTIME to prevent these types of queries:

- A long-running query against a database that accesses atrributes at a high level, forcing many dynamic calculations to occur.
- A query that uses the "Drill to bottom" option in a large dimension.
- A query that uses the "Drill to all levels" option in a a large dimension.

Use QRYGOVEXECTIME, for example, if you have users who try to retrieve so much data in a single query that their query appears to hang for minutes at a time.

Notes

- Because the query time setting is evaluated in 10 second increments, the query may actually run nine seconds longer than specified before being terminated.
- If you use an invalid value (such as a negative number, a letter, a word, or a special character) for *n*, the server ignores QRYGOVEXECTIME.
- Query governor settings are ignored during data load and calculation. You can leave query governor settings in the configuration file whether you are performing these operations or querying against the data.

Example

```
QRYGOVEXECTIME Sample Basic 20
```

Sets 20 seconds as the maximum time that a query can run before being terminated. In this example the restriction applies only to the Basic database in the Sample application.

```
QRYGOVEXECTIME 45
```

Sets 45 seconds as the maximum time that a query can run before being terminated. The query time limit applies to all applications and databases on the server.

See Also

QRYGOVEXECBLK

QUERYTRACE

Sets a query calculation flow trace to be run and the results to be printed out to a file.

Description

This setting enables query tracing for calculation flows. The query tracing output file includes:

- The input query
- An expanded query odometer
- General information about query calculation units
- A list of formulas and aggregations
- An ordered list of all output cells that are calculated or aggregated during the query, according to solve order

Notes

- This setting applies to block storage and hybrid aggregation databases.
- The query tracing output file, `query_trace.txt`, is written to the database files location.

Syntax

```
QUERYTRACE n
```

Where *n* should be set to `-1`, to enable query tracing.

When to Use Different Query Tracing Options

The QUERYTRACE configuration setting is designed for debugging a single query, in case any problems are observed. It prints the trace log to the cube directory as `query_trace.txt`, and the file is cleared by default before each query execution. QUERYTRACE should not be left enabled in the application configuration for long term use, as it may affect performance.

The TRACE_REPORT configuration setting provides fewer details than QUERYTRACE, but is able to trace multiple, concurrent queries. TRACE_REPORT can be enabled for long term use, as it does not affect performance. It is a good option for gathering information on many concurrent queries running over a period of time. TRACE_REPORT prints the trace log to the cube directory as `trace_report.log`, and the file is not cleared between queries. If LONGQUERYTIMETHRESHOLD is enabled, this setting is disabled, and nothing is printed to `trace_report.log`.

The LONGQUERYTIMETHRESHOLD configuration setting is a good choice for production environments. It prints statistics to the application log file about any query that runs longer than a defined number of milliseconds.

Enabling query logging by setting the QUERYLOG parameter in *dbname*`.cfg` (in the cube directory) is designed for tracking user query patterns for a cube, in XML format.

Example

```
QUERYTRACE -1
```

Sets a tracing query to be run that includes all tracing features listed in Description.

See Also

QUERYTRACETHRESHHOLD

QUERYTRACETHRESHOLD

Sets the maximum number of cells than can be displayed for each formula or aggregation number for query tracing for calculation order analysis.

Description

This setting specifies the maximum number of cells (or tuples) that a QUERYTRACE query will display for an MDX query or grid client query. Use this setting to limit the number of cells printed for each formula or aggregation number.

Notes

This setting applies to block storage and hybrid aggregation databases.

Syntax

```
QUERYTRACETHRESHOLD n
```

Where *n* is an integer value between 0 and unlimited, specifying the maximum number of cells to display for each formula calculation path. The default value is 100.

Example

```
QUERYTHRESHOLD 50
```

Sets 50 as the maximum number of cells to be displayed for a query tracing the solve order or calculation order.

See Also

QUERYTRACE

RENEGADELOG

Enables logging of members loaded into a renegade member intersection.

By default, Essbase does not create a log file to track data loaded to renegade members. If RENEGADELOG is set to true, Essbase creates a log file in the Essbase logs directory. The log file name is `renDataload_filename`*`filename_timestamp`*`.log` for non-SQL data loads and `renDataload_SQL_`*`timestamp`*`.log` for SQL-based data loads.

The log file records the data value loaded to the renegade member. If more than one member in a given data load is missing for a dimension with renegade members enabled, the log file lists only one value. Information on the remaining missing data values is provided in comments.

Syntax

```
RENEGADELOG [appname [dbname]] TRUE | FALSE
```

- *appname*—Application name. Optional parameter for applying the TRUE or FALSE setting to one or all databases within the application. If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application. If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Database name. Optional parameter for applying the TRUE or FALSE setting to the specified database within the specified application. If you do not specify a value for *dbname*, the setting applies to all databases within the specified application. If *appname* is not specified, you cannot specify *dbname*.
- TRUE—Creates a log file to track data loaded to renegade members.
- FALSE—No log file is created. This is the default value.

Example

```
RENEGADELOG TRUE
```

RESTRUCTURETHREADS

Specifies whether parallel restructuring is enabled for a database and the number of threads to use.

This setting does not apply to aggregate storage databases.

Syntax

```
RESTRUCTURETHREADS [ appname [ dbname] ] n
```

- *appname*—Application name. Optional parameter for enabling parallel restructuring for one or all databases in an application.
- *dbname*—Database name. Optional parameter for enabling parallel restructuring for an individual database. This parameter must be used in combination with *appname*.
- *n*—Number of threads to use in parallel restructuring.

Notes

- Use the value `xxxxx` to indicate "all" for any application or database argument. For example:

  ```
  RESTRUCTURETHREADS xxxxx Basic 2
  ```

  enables parallel restructuring for any application with a Basic database.

- Settings for nonexistent applications or databases are ignored.
- If RESTRUCTURETHREADS is not defined, the default is one thread.
- Oracle recommends setting an application's RESTRUCTURETHREADS to 2 for most systems, or 4 if the application runs on Exalytics. Check your calculation and restructure performance after making any changes.

Description

This setting specifies whether parallel restructuring is enabled for a database and the number of threads to use. You can enable parallel restructuring for individual databases, or for all databases in an application. For more information about parallel restructuring, see Parallel Restructuring in *Designing and Maintaining Essbase Cubes*.

Examples

```
RESTRUCTURETHREADS Sample 2
```

Specifies two threads and applies to all databases in the Sample application

```
RESTRUCTURETHREADS Sample Basic 4
```

Specifies four threads and applies to the Basic database in the Sample application

RTDEPCALCOPTIMIZE

Sets whether the @CURRMBRRANGE calculation function behaves as runtime dependent or non runtime dependent.

Syntax

```
RTDEPCALCOPTIMIZE [appname [dbname]] TRUE | FALSE
```

- *appname*—Optional. If you specify an application name, the setting applies to all databases within the named application. If you do not specify an application name, the setting applies to all applications and databases on the Essbase Server.
- *dbname*—Optional. If you specify a database name and an application name, the setting applies only to the named database. If you do not also specify an application name, the database is ignored and the setting applies to all applications and databases on the Essbase Server.
- TRUE—This is the default. @CURRMBRRANGE behaves as a non runtime dependent formula. This, the default behavior, could result in incorrect calculation results if the @CURGEN or @CURLEV functions are used as arguments to @CURMBRRANGE, because Essbase would fail to generate the correct dependency list to compute @CURRMBRRANGE.
- FALSE—@CURRMBRRANGE behaves as runtime dependent formula, but only when @CURGEN or @CURLEV are passed as an argument to

@CURRMBRRANGE. Calculations involving @CURRMBRRANGE may run slowly, as computation of runtime dependent formulas requires more memory.

Example

```
RTDEPCALCOPTIMIZE FALSE
```

SERVERTHREADS

Overrides the default value of the number of threads that the application process (ESSSVR) can spawn. Application threads are used in calculations, client requires, administrative activities, etc.

When a transaction is requested, the application process (ESSSVR) assigns a thread to the transaction and releases the thread when the transaction is completed.

Syntax

```
SERVERTHREADS [appname] n
```

- *appname*—Optional. Specifies an application; the SERVERTHREADS setting applies to all databases within the named application.

  If you do not specify an application, the setting applies to all applications and databases on the Essbase instance.
- *n*—Specifies the number of threads that the application process (ESSSVR) can spawn; 20 to 1024, inclusive.

  The default value is 20.

  If you specify a value that is:

  – Less than the minimum, Essbase interprets the value as 20

  – Greater than the maximum, Essbase interprets the value as 1024

Notes

- While the actual maximum value you can set is 1024, the maximum number of threads an operating system can handle might be much lower. Before specifying a value greater than the default value, check with your system administrator, as higher values can significantly consume system resources.
- If the computer on which Essbase Server runs freezes while running multiple reports simultaneously, increase the value of SERVERTHREADS by one for each report you run.
- Each application thread may create child threads for tasks such as parallel calculation, parallel data load or export, and parallel restructuring. If the total number of running threads is too high, threads may lose efficiency in contending for server resources.

Example

```
SERVERTHREADS 25
```

Allows all applications on Essbase Server to spawn up to 25 threads.

```
SERVERTHREADS Sample 100
```

Allows the Sample application on Essbase Server to spawn up to 100 threads.

SSANCESTORONTOP

Controls whether users can specify that ancestors be positioned at the top, in grid client operations.

Syntax

```
SSANCESTORONTOP [appname [dbname]] TRUE | FALSE
```

- *appname*—Optional. Specifies the application to which the configuration applies.
- *dbname*—Optional. Specifies the cube to which the configuration applies.
- TRUE—Smart View users can specify the ancestor position for hierarchies in ad hoc grids.
- FALSE—Smart View users cannot specify the ancestor position. This is the default.

Description

If this configuration property is set to TRUE, Smart View users can specify ancestor position for hierarchies in ad hoc grids. By default, this parameter is not enabled, and the ancestor is positioned at the bottom.

Example

```
SSANCESTORONTOP Sample TRUE
```

See Also

Specifying Ancestor Position in Ad Hoc Grids in *Working with Oracle Smart View for Office* (available on the Books tab)

SSMEMBERIDPROCESSING

Controls whether Smart View keeps track of members in a report by using stable member IDs instead of (less stable) uniquely qualified member names.

For a database that has duplicate member names enabled, an internal member ID is associated with each member. However, member IDs are only applicable for cubes deployed from Essbase Business Intelligence Acceleration Wizard.

Syntax

```
SSMEMBERIDPROCESSING [appname [dbname]] TRUE | FALSE
```

- *appname*—Optional. Specifies the application for which member IDs should be used.

  If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

  To enable the setting for a specific database, you must specify an application and database.

  If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on Essbase Server.
- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which member IDs should be used.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- TRUE—Essbase tracks members using stable member IDs. This is the default for BI outlines, if DISPLAY_KEY and MEMBER_VALUE alias tables exist in the outline.
- FALSE—Essbase tracks members using qualified member names.

Description

For Smart View reports on duplicate member name outlines, member IDs can help Smart View maintain report validity for all members, even when members in the outline are moved or renamed.

Notes

If you opt to track members using qualified member names instead of member IDs, Smart View reports may become invalid if members in the outline are moved or renamed.

Example

```
SSMEMBERIDPROCESSING Sample TRUE
```

SSOPTIMIZEDGRIDPROCESSING

Specifies whether optimized grid processing, which cuts the input grid into symmetric grids to create fewer symmetric queries, is enabled for grid client operations.

Syntax

```
SSOPTIMIZEDGRIDPROCESSING [appname [dbname]] TRUE | FALSE
```

- *appname*—Optional. Specifies the application for which optimized grid processing is to be set.

If you specify a value for *appname* and do not specify a value for *dbname*, the setting applies to all databases in the specified application.

To enable the setting for a specific database, you must specify an application and database.

If you do not specify an application, you cannot specify a database, and the setting applies to all applications and databases on Essbase Server.

- *dbname*—Optional. Specifies the database, in the application specified by *appname*, for which optimized grid processing is to be set.

  If you specify a value for *dbname* but do not specify a value for *appname*, your specification is ignored.
- TRUE—Enables optimized grid processing for grid client operations.

  The default value is TRUE.
- FALSE—Disables optimized grid processing for grid client operations.

For changes to the configuration file to take effect, you must restart Essbase Server.

Example

```
SSOPTIMIZEDGRIDPROCESSING FALSE
```

Turns off optimized processing for grid client operations on all applications and databases on Essbase Server.

SSPROCROWLIMIT

Controls the maximum number of rows Essbase processes on a Smart View or other grid client request.

Syntax

```
SSPROCROWLIMIT n
```

n—An integer value of 16,384 or higher. The default value is 250,000.

Description

SSPROCROWLIMIT controls the maximum number of rows Essbase processes on a Smart View or other grid client user request. SSPROCROWLIMIT is in effect only for grid clients when the Suppress #Missing Rows option is selected The rows are counted before suppression; that is, missing rows and rows containing zero values are included.

When users zoom in on one or more members, Essbase must process a larger grid containing selected members expanded to the zoom-in level set in the options. When the Suppress #Missing Rows option is set, Essbase returns only rows with at least one column containing a non-missing value. SSPROCROWLIMIT defines the maximum size (number of rows) of the larger grid that Essbase needs to process. This setting prevents excessive memory usage for a single grid operation.

When the Excel Suppress #Missing Rows option is not selected, the limit is 64000.

Notes

- SSPROCROWLIMIT applies to unprocessed rows; that is, it is the number of rows Essbase accepts before processing. Row processing eliminates missing rows. After processing, the number of rows that the client can retrieve depends on grid-client-defined limits.
- If SSPROCROWLIMIT is exceeded, Essbase issues an error message and stops processing the request.
- This setting is not used in the Smart View Free form mode.
- Oracle does not recommend using a limit higher than 500,000.

Example

```
SSPROCROWLIMIT 300000
```

SUPNA

Controls whether the Suppress #Missing Rows option in Smart View or another grid client interface suppresses the display of cells for which a user has no access (in addition to suppressing `#MISSING` rows).

Syntax

```
SUPNA ON | OFF
```

- ON—The Suppress #Missing Rows option suppresses the display of cells for which a user has no access.
- OFF—The Suppress #Missing Rows option does not suppress the display of cells for which a user has no access. This is the default.

Description

The Suppress #Missing Rows option in Smart View or other grid clients suppresses the display of data rows that contain only missing values. SUPNA specifies whether Essbase also suppresses the display of cells for which a user has no access.

Example

```
SUPNA OFF
```

For all databases on the server, Essbase does not suppress cells for which a user has no access. These cells appear in the grid as #NoAccess. Rows of missing data are suppressed.

SVRIDLETIME

Sets the number of minutes an Essbase application can be idle before it is shut down.

If an application is idle for the time period set here, it will stop running. When any new request is made (for example, Smart View query, Outline, or CLI activity), the application starts automatically.

Syntax

```
SVRIDLETIME [appname] n
```

Where *n* is the number of minutes of idle time permitted before shutdown. The default value is 120 minutes. The minimum value is 1 minute. The maximum value is 20160 minutes (two weeks). To disable automatic shutdown, set *n* to 0.

Notes

- By default, applications are set to shut down after 2 hours of idle time.
- The SVRIDLETIME value should always be set to a few more minutes than the idle user session logout interval, which you can set using the MaxL alter system **set session_idle_limit** grammar.

Example

```
SVRIDLETIME Sample 15
```

TARGETASOOPT

Potentially optimizes large queries (from Smart View or other grid clients, MDX, or Report Writer) to an aggregate storage database across a transparent partition when the source outline and target outline are identical in the partition region definition area.

Syntax

```
TARGETASOOPT [appname] TRUE | FALSE
```

- *appname*—Optional. Application name. If you specify a value for *appname*, the setting applies to all databases in the specified application. If you do not specify an application, the setting applies to all applications and databases on the Essbase Server.
- FALSE—The default. Optimization is not enabled, even if queries match the required criteria (see Description).
- TRUE—Optimization is enabled for queries that match the required criteria (see Description).

  When TARGETASOOPT is TRUE, Essbase completes the following steps:

  1. When the partition is next validated, automatically determines if the partition region definition outlines are identical on the source and target databases

2. If the partition region definition outlines are identical, the query is sent in the compact format from the target database to the source database.

You must restart Essbase Server to initialize any change to the configuration file.

Description

TargetASOOpt enables an alternate (compact) format for sending a query (from Smart View or other grid clients, MDX, or Report Writer) to an aggregate storage source database, and hence may speed up large queries between databases that match the following criteria:

- Databases are transparently partitioned (for example, to enable write-back for aggregate storage databases)
- Source is an aggregate storage database
- Partitioned area definitions in the source and target are identical (for example in the Sample Basic database, if the partition region definition is@idesc("100"), then the outline hierarchies below Time, Market, Measures, Scenario, and 100, must be identical on the source and target databases)
- Source outline and target outline are identical

Notes

If at query time the source and target outlines have been modified after the last validation, even if the partition region definition outlines are still identical, TARGETASOOPT is dissabled for the query. To enable TARGETASOOPT for the query, you must revalidate the partitions.

Example

```
TARGETASOOPT TRUE
```

See Also

TARGETTIMESERIESOPT

TARGETTIMESERIESOPT

Globally sets query optimization across transparent partitions for outlines that have a time dimension with Dynamic Time Series members. If this setting is specified, queries with Dynamic Time Series members will incur faster query times. Use this setting only if the time dimensions on the source and target partitions are identical. If the time dimensions on the source and target partitions are not the same, this setting may produce incorrect results. Restart Essbase to enable this setting to take effect for the Dynamic Time Series members that have been enabled at run time.

Syntax

```
TARGETTIMESERIESOPT TRUE | FALSE
```

- TRUE—Enables query optimization across transparent partitions for outlines that have a time dimension with Dynamic Time Series members.
- FALSE—Query optimization is not enabled. This is the default.

Example

```
TARGETTIMESERIESOPT TRUE
```

See Also

TARGETASOOPT

TRACE_REPORT

Enables query tracing, and prints results to a file. Can trace multiple queries running at the same time.

Description

This setting enables query tracing for grid retrievals, MDX queries, and transparent partition sources. The query tracing output file includes information on:

- The query type
- The time of the query
- User name and IP address
- Rows and columns processed

When to Use Different Query Tracing Options

The QUERYTRACE configuration setting is designed for debugging a single query, in case any problems are observed. It prints the trace log to the cube directory as `query_trace.txt`, and the file is cleared by default before each query execution. QUERYTRACE should not be left enabled in the application configuration for long term use, as it may affect performance.

The TRACE_REPORT configuration setting provides fewer details than QUERYTRACE, but is able to trace multiple, concurrent queries. TRACE_REPORT can be enabled for long term use, as it does not affect performance. It is a good option for gathering information on many concurrent queries running over a period of time. TRACE_REPORT prints the trace log to the cube directory as `trace_report.log`, and the file is not cleared between queries. If LONGQUERYTIMETHRESHOLD is enabled, this setting is disabled, and nothing is printed to `trace_report.log`.

The LONGQUERYTIMETHRESHOLD configuration setting is a good choice for production environments. It prints statistics to the application log file about any query that runs longer than a defined number of milliseconds.

Enabling query logging by setting the QUERYLOG parameter in *dbname*`.cfg` (in the cube directory) is designed for tracking user query patterns for a cube, in XML format.

Notes

- This setting applies to block storage and hybrid mode databases.
- The query tracing report file, `trace_report.log`, is written to the database files location.
- If LONGQUERYTIMETHRESHOLD is set, TRACE_REPORT is not active (nothing will be printed to `trace_report.log`).

Syntax

```
TRACE_REPORT n
```

Where *n* should be set to `-1`, to enable query tracing.

Example

```
TRACE_REPORT -1
```

Sets a tracing query to be run that includes all tracing features listed in Description.

Configuration Settings Categorical List

This section lists all of the configuration settings, grouped categorically. Some may appear in more than one category.

- Calculation Configuration Settings
- Data Import and Export Configuration Settings
- Logging and Error Handling Configuration Settings
- Memory Management Configuration Settings
- Miscellaneous Configuration Settings
- Partitioning Configuration Settings
- Ports and Connections Configuration Settings
- Query Management Configuration Settings

Calculation Configuration Settings

- ASODYNAMICAGGINBSO
- ASODYNAMICAGGINBSOFOLDERPATH
- CALCCACHE
- CALCCACHEHIGH
- CALCCACHEDEFAULT
- CALCCACHELOW
- CALCLIMITFORMULARECURSION
- CUSTOMCALCANDALLOCTHRUINSERT
- DYNCALCCACHEMAXSIZE
- FORCEALLDENSECALCON2PASSACCOUNTS
- HYBRIDBSOINCALCSCRIPT
- RTDEPCALCOPTIMIZE

Data Import and Export Configuration Settings

- DLSINGLETHREADPERSTAGE
- DLTHREADSPREPARE
- DLTHREADSWRITE

Memory Management Configuration Settings

- DATACACHESIZE
- DYNCALCCACHEMAXSIZE
- INDEXCACHESIZE
- MAXFORMULACACHESIZE
- NUMBLOCKSTOEXTEND
- SSOPTIMIZEDGRIDPROCESSING
- SSPROCROWLIMIT

Logging and Error Handling Configuration Settings

- ENABLERTSVLOGGING
- GRIDEXPANSIONMESSAGES

Miscellaneous Configuration Settings

- AUTOMERGE
- AUTOMERGEMAXSLICENUMBER
- MAXNUMBEROFACTIVEDB
- NUMBLOCKSTOEXTEND
- RESTRUCTURETHREADS
- TARGETTIMESERIESOPT

Partitioning Configuration Settings

- MAX_REQUEST_GRID_SIZE
- MAX_RESPONSE_GRID_SIZE

Ports and Connections Configuration Settings

- AGENTTHREADS
- MAXLOGINS
- SERVERTHREADS

Query Management Configuration Settings

- ASODYNAMICAGGINBSO
- ASODYNAMICAGGINBSOFOLDERPATH
- GRIDEXPANSION
- GRIDEXPANSIONMESSAGES
- GRIDSUPPRESSINVALID
- HYBRIDBSOINCALCSCRIPT
- IGNORECONSTANTS
- LONGQUERYTIMETHRESHOLD
- LONGREQTIMETHRESHOLD
- MAXFORMULACACHESIZE
- MDXQRYGOVCOUNT
- QRYGOVEXECBLK
- QRYGOVEXECTIME
- QUERYRESULTLIMIT
- QUERYTRACE
- QUERYTRACETHRESHOLD
- SSANCESTORONTOP
- SSOPTIMIZEDGRIDPROCESSING
- SSPROCROWLIMIT
- SUPNA
- TARGETASOOPT
- TRACE_REPORT

See also Query Logging Configuration, which you can enable by means of a separate configuration file.

Aggregate Storage and Block Storage Settings Comparison

Some configuration settings apply only to block storage databases, some apply only to aggregate storage databases, and some are applicable to both.

Topics:

- Block Storage and Aggregate Storage Configuration Settings
- Aggregate Storage Configuration Settings
- Block Storage Configuration Settings

Block Storage and Aggregate Storage Configuration Settings

The following settings apply to aggregate storage databases and to block storage databases.

- AGENTTHREADS
- DLSINGLETHREADPERSTAGE
- DLTHREADSPREPARE
- GRIDEXPANSION
- GRIDEXPANSIONMESSAGES
- GRIDSUPPRESSINVALID
- MAXLOGINS
- MAXNUMBEROFACTIVEDB
- QRYGOVEXECTIME
- SERVERTHREADS
- SSANCESTORONTOP
- SSMEMBERIDPROCESSING
- SSOPTIMIZEDGRIDPROCESSING
- SSPROCROWLIMIT
- SUPNA
- SVRIDLETIME
- TARGETASOOPT
- TARGETTIMESERIESOPT

Aggregate Storage Configuration Settings

The following settings apply only to aggregate storage databases.

- AUTOMERGE
- AUTOMERGEMAXSLICENUMBER
- CUSTOMCALCANDALLOCTHRUINSERT
- DEFAULTVIEWBUILD
- DEFAULTVIEWBUILDSIZE
- MAX_REQUEST_GRID_SIZE
- MAX_RESPONSE_GRID_SIZE

Block Storage Configuration Settings

The following settings apply only to block storage databases.

- ASODYNAMICAGGINBSO
- ASODYNAMICAGGINBSOFOLDERPATH

- AUDITTRAIL
- CALCCACHE
- CALCCACHEHIGH
- CALCCACHEDEFAULT
- CALCCACHELOW
- CALCLIMITFORMULARECURSION
- CALCTRACE
- DATACACHESIZE
- DLTHREADSWRITE
- DYNCALCCACHEMAXSIZE
- ENABLERTSVLOGGING
- FORCEALLDENSECALCON2PASSACCOUNTS
- HYBRIDBSOINCALCSCRIPT
- INDEXCACHESIZE
- QRYGOVEXECBLK
- RESTRUCTURETHREADS

3
Provider Services Configuration

The Provider Services configuration properties are available in the Essbase web interface to help you manage network timeout parameters.

You can set the values for Provider Services configurations in the Console. See Set Provider Services Configuration Properties.

essbase.jobs.maxCount

Sets the maximum number of parallel jobs that may be run.

Syntax

```
essbase.jobs.maxcount n
```

n—Specifies the number of active jobs that Essbase can run in parallel. The default is 10.

Description

Jobs are operations you can run from the Essbase web interface. Jobs include loading data, building dimensions, exporting cubes, running MaxL scripts, running calculations, and clearing data. Jobs are asynchronous, meaning they are run in the background as a unique thread. Each job has a unique id.

olap.server.netRetryCount

Specifies the number of times an API can retry a unsuccessful network operation before failing and reporting an error. If olap.server.netSocketTryInfinite is set to true, then olap.server.netRetryCount is ineffective.

Syntax

```
olap.server.netRetryCount n
```

n—An integer value. The default is 600 retries. Every five tries gives a timeout of one second. The default value, 600, gives a timeout of two minutes.

olap.server.netSocketTryInfinite

Specifies whether the client will keep trying infinitely on a network operation. If set to true, then olap.server.netRetryCount is ineffective.

Syntax

```
olap.server.netSocketTryInfinite true|false
```

4

Query Logging Configuration

Query Logging Overview

Query logging provides a way for Essbase administrators to track query patterns of an Essbase database. The query log file tracks queries performed against the database from Smart View, Report Writer, or Grid-API clients. Query logging can track generation or level numbers of members belonging to specific generations or levels. Query logging also offers the flexibility to exclude logging of certain dimensions and members belonging to certain generations or levels. Because the query log file output is an XML document, you can import the log file to any XML-enabled tool to view the log.

> **Note:**
>
> You can import the .XML file to Microsoft Access or Microsoft Excel. However, you must first shut down the database.

For details about the query log file structure, refer to `querylog.dtd` in the *ARBORPATH*`/bin` directory.

Query logging is available for both block storage and aggregate storage databases.

To enable query logging, create a query log file and add to the file the settings that control how query logging is performed.

You must create a query log file for each database that requires query logging. If the query log file is missing or the QUERYLOG setting is off, query logging is disabled.

Query Logging Settings Procedure

The following steps explain how to create a query log settings file. To see a sample query log file, see Query Logging Sample File.

To enable query logging:

1. In the *ARBORPATH*`\App\`*appname*`\`*dbname* directory of Essbase, create a query log settings file.

The settings file must be named *dbname*.cfg, where *dbname* matches the name of the database. For example, the query log settings file for Sample Basic is basic.cfg. For databases in Unicode-mode applications, the query log file must be encoded in UTF-8 and include the UTF-8 signature.

2. In the settings file, specify required and optional elements, using the syntax from the section Query Logging Syntax:
   - The dimension for which you want to log queries (QUERYLOG [*dimension_name*]).
   - **Optional:** The setting to log generation or level numbers for members of specified generations or levels in a dimension (QUERYLOG GENERATION *generation-range* or QUERYLOG LEVEL *level-range*).
   - **Optional:** The setting to exclude logging of members from specified generations or levels in a dimension (QUERYLOG NONE GENERATION *generation-range* or QUERYLOG NONE LEVEL *level-range*).
   - **Optional:** The location where the query log file is created (QUERYLOG LOGPATH *path-expression*).
   - **Optional:** The format of the log file output (QUERYLOG LOGFORMAT CLUSTER | TUPLE).
   - **Optional:** The size of the log file (QUERYLOG LOGFILESIZE *n*)
   - **Optional:** The size of all log files (QUERYLOG TOTALLOGFILESIZE *n*).
   - A setting to enable or disable query logging the next time the application starts (QUERYLOG ON | OFF).
3. Restart the database to accept the settings.

> **Note:**
>
> Restart after creating a file or changing any entries in a file.

4. After query logging is enabled, review the log entries in the query log file, *dbname*.qlg.

   For example, you can view the output of the log file to analyze how many times a certain member has been queried. You can use a UTF-8-enabled editor to view query log files for databases in Unicode-mode applications.

Query Log Settings File Syntax

The query log settings filename must be of the form *dbname*.cfg, where *dbname* represents the name of a database. The *dbname*.cfg file must be located in the ARBORPATH\App*appname**dbname* directory of Essbase. The *dbname*.cfg file consists of the following syntax:

```
QUERYLOG [dimension_name]
QUERYLOG NONE GENERATION generation-range
QUERYLOG NONE LEVEL level-range
QUERYLOG GENERATION generation-range
QUERYLOG LEVEL level-range
QUERYLOG LOGPATH path-expression
```

```
QUERYLOG LOGFORMAT CLUSTER | TUPLE
QUERYLOG LOGFILESIZE n
QUERYLOG TOTALLOGFILESIZE n
QUERYLOG ON | OFF
```

Table 4-1 QUERYLOG Parameters

QUERYLOG Parameter	Description
[*dimension_name*]	Identifies the dimension name to be tracked. The brackets around the dimension name are required. QUERYLOG [*dimension_name*] logs all members of a dimension. For example, QUERYLOG [Product] tracks all members of the Product dimension. Each dimension must be specified in a separate QUERYLOG [*dimension_name*] setting.
NONE GENERATION generation-range	Prevents tracking of members from the specified generation range. For example, QUERYLOG NONE GENERATION 2 excludes tracking of all members from generation 2 of the named dimension.
NONE LEVEL level-range	Prevents tracking of members from the specified level range. For example, QUERYLOG NONE LEVEL 0-2 excludes tracking of all members of levels 0, 1, and 2 of the named dimension.
GENERATION generation-range	Tracks members of the specified generation range by generation number, rather than by member name. For example, QUERYLOG GENERATION 5-7 logs members of generations 5, 6, and 7 of the named dimension by their generation number in the log file.
LEVEL level-range	Tracks members of the specified level range by level number, rather than by member name. For example, QUERYLOG LEVEL -3 logs members of levels 0, 1, 2, and 3 of the named dimension by their level number in the log file.
LOGPATH *path-expression*	Specifies the location of the output log file. The log file name is *dbname*`00001.qlg`; for example, `basic00001.qlg`. Examples of the log path are QUERYLOG LOGPATH `/usr/local/Essbaselogs/` and QUERYLOG LOGPATH `d:\Essbaselogs\querylogs\`. You must include a backslash \ (for Windows directories) or forward slash / (for UNIX directories) at the end of the path expression; otherwise, the query log file is not created. By default, the location for the log output file is the *ARBORPATH*`\App\`*appname*`\`*dbname*`\` directory. If the LOGPATH *path-expression* setting is missing, the default is used. Essbase writes log information to the query log file after an application stops running.

Table 4-1 (Cont.) QUERYLOG Parameters

QUERYLOG Parameter	Description
LOGFORMAT CLUSTER | TUPLE	Specifies the format of the log output. CLUSTER and TUPLE provide the same log information, but display the information differently. CLUSTER provides information on how many members of a dimension were queried and lists queried members within their respective dimensions. TUPLE lists each queried member combination. By default, CLUSTER is the log format. Because the TUPLE format lists each member combination queried, TUPLE may have a greater impact on query performance than CLUSTER. See Sample Cluster Output for an example of a query log in cluster format. See Sample Tuple Output for an example of a query log in tuple format.
LOGFILESIZE *n*	Specifies the maximum size of an individual query log file in megabytes (MB). The minimum value is 1 MB. The maximum value is 2048 MB (2 GB). If the LOGFILESIZE setting is missing, then, by default, the query log file size is 1 MB. If an initial query log file size exceeds the specification, log information is added to a new query log file. Each time a new file is created, the filename is incremented by one.
TOTALLOGFILESIZE *n*	Specifies the maximum size of all query log files combined in megabytes (MB). The minimum value is 512 MB (1/2 GB). The maximum value is 4095 MB. If the TOTALLOGFILESIZE setting is missing, then, by default, the total query log file size is 1024 MB (1 GB). Query log files are created until the file size total exceeds the specified maximum. When the maximum is exceeded, a message is displayed and query logging automatically turns off.
ON | OFF	Specifies whether the query logging feature is turned on or off. All query log settings are ignored if this setting is OFF or missing. By default, the setting is OFF.

Generation-range and *level-range* values are represented in one of the following ways:

Table 4-2 Generation and Level Range Specifications

Generation-Range or Level-Range Value	Description
x	A specific generation or level number. For example, QUERYLOG NONE GENERATION 2 excludes generation 2 from query logging.

Table 4-2 (Cont.) Generation and Level Range Specifications

Generation-Range or Level-Range Value	Description
x-y	All generations or levels inclusive of number x through number y. For example, QUERYLOG GENERATION 1-3 or QUERYLOG LEVEL 1-3 includes generation or level numbers 1, 2, and 3.
-x	For *generation-range*, all generations within the range 1 through *x*. For *level-range*, all levels within the range 0 through *x*. For example, QUERYLOG GENERATION -2 includes generations 1 and 2. QUERYLOG LEVEL -3 includes levels 0, 1, 2, and 3.
x-	For *generation-range*, all generations within the range from number *x* through the highest generation. For *level-range*, all levels within the range from number *x* through the highest level. For example, QUERYLOG Level 1- includes levels 1, 2, 3 and so on up to the highest level.

Notes

- When query logging is enabled, queries to the database may be slower. Performance depends on how many members are being tracked and the size of the query.
- If the settings file name does not match the name of the database or the settings file is located in a place other than the *ARBORPATH*\App*appname**dbname* directory, Essbase ignores query logging.
- If, in the settings, QUERYLOG ON is missing or if QUERYLOG OFF is set, query logging is disabled.
- If generation and level settings cause contradictions in the settings file, the following precedence rules apply:
  - generation numbers (highest priority)
  - level numbers
  - member names (lowest priority)

  For example, if a member belongs to both level 1 and generation 2 and the settings QUERYLOG GENERATION 2 and QUERYLOG NONE LEVEL 1 are in the settings file, the generation setting takes precedence, and members of generation 2 are logged by generation number.

Tips

- To view query log output easily, change the file extension `.QLG` to `.XML`, and then using the Internet Explorer or Netscape browser view the `.XML` file.

> **Note:**
>
> You can import the .XML file to Microsoft Access or Microsoft Excel. However, you must first shut down the database.

- If Essbase is not producing a query log file as expected, view the *dbname*`.log` file in the *ARBORPATH*`\App\`*appname* directory to search for query log messages.

Query Logging Sample File

> **Note:**
>
> # indicates a comment that describes a line of the settings file. Comments are not necessary to include in the actual query log settings file.

```
# Log the Product dimension
QUERYLOG [Product]
# Log the Market dimension
QUERYLOG [Market]
# Log members of generation 2 of Market by generation number
QUERYLOG GENERATION 2
# Display log output in cluster format
QUERYLOG LOGFORMAT CLUSTER
# Create log file in C:\QUERYLOG\
QUERYLOG LOGPATH C:\QUERYLOG\
# Start a new log file after an individual log file size reaches 2 MB
QUERYLOG LOGFILESIZE 2
# Turn off query logging after the total size of all log files reaches
1024 MB (1 GB)
QUERYLOG TOTALLOGFILESIZE 1024
# Enable query logging
QUERYLOG ON
```

Query Logging Sample Output

The following seSample Query Log Outputgment shows an example of how log settings look in a log file. In the example, the log settings show that all members of Product are logged and that members of generation 2 of Market are logged by generation number. The log format is cluster and the log path is `C:\QUERYLOG\`.

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <root>
  - <session>
     <bootuptime>Wed Jul 23 15:27:26 2002</bootuptime>
   - <logsettings>
    - <dimensions>
     - <logdim name="Product">
```

```
    - <logdim name="Market">
       <spec>GENERATION 2</spec>
     </logdim>
    </dimensions>
  - <othersettings>
    <logformat>cluster</logformat>
    <logpath>C:\QUERYLOG\</logpath>
   </othersettings>
   </logsettings>
```

Description

A query is a unit of retrieval from the user perspective. The way a user may perceive a query is different than how the server analyzes and executes a query. Even if a user performs a single retrieval, in order for the server to efficiently execute the logical query, the server splits the query into a number of subqueries to execute. Therefore, a single retrieval from the user perspective may actually consist of several subqueries from the server perspective. These subqueries are reflected in the query log.

Sample Cluster Output

The following segment shows an example of how queries are logged in cluster format. The username is listed along with the query execution date and the start time of the query. Each cluster contains two dimension entries. The first cluster shows that members 100 and 200 of the Product dimension were queried. The second cluster shows that member 300 of Product and Generation 2 of Market were queried. The elapsed time to perform the query is also provided.

```
<query>
 <user>User1</user>
 <time>Tue Aug 13 12:29:49 2002</time>
 <subquery>
  <cluster size="2">
   <dim size="2">
    <member>100</member>
    <member>200</member>
   </dim>
   <dim size="1">
    <member>Market</member>
   </dim>
  </cluster>
 </subquery>
 <subquery>
 <cluster size="2">
  <dim size="1">
   <member>300</member>
  </dim>
  <dim size="2">
   <member>Market</member>
   <generation>2</generation>
  </dim>
 </cluster>
 </subquery>
 <elapsedtime>0.016 seconds</elapsedtime>
</query>
```

Sample Tuple Output

The following segment shows an example of how queries are logged in tuple format. The username is listed along with the query execution date and the start time of the query. Note that each member of Product is displayed with Market. Each possible member combination is displayed for a given query. The elapsed time to perform the query is also provided.

```
<query>
 <user>User1</user>
 <time>Tue Aug 13 12:28:14 2002</time>
 <subquery>
  <tuples>
   <tuple>
     <member>100</member>
     <member>Market</member>
   </tuple>
  </tuples>
 </subquery>
 <subquery>
  <tuples>
   <tuple>
    <member>200</member>
    <member>Market</member>
   </tuple>
  </tuples>
 </subquery>
 <elapsedtime>0.02 seconds</elapsedtime>
</query>
```